artment
ortation
al Highway
Safety
Administration

DOT-HS-807-158
DOT-TSC-NHTSA-86-6
Final Report

December 1987

Analysis of Head Response to Torso Acceleration

Vol. II - Description of Data Retrieval, Analysis and Display Software

C.H. Spenny

Transportation Systems Center
Cambridge, MA 02142

Prepared for

National Highway Traffic Safety Administration
Research and Development
Washington, DC 20590

Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
DOT-HS-807-158		

4. Title and Subtitle	5. Report Date
Analysis of Head Response to Torso Acceleration Vol. II - Description of Data Retrieval, Analysis and Display Software	December 1987
	6. Performing Organization Code DTS-44
7. Author(s) C.H. Spenny*	8. Performing Organization Report No. DOT-TSC-NHTSA-86-6
9. Performing Organization Name and Address U.S. Department of Transportation Research and Special Programs Administration Transportation Systems Center Cambridge, MA 02142	10. Work Unit No. (TRAIS) HS476/R4434
	11. Contract or Grant No.
12. Sponsoring Agency Name and Address U.S. Department of Transportation National Highway Traffic Safety Administration Research and Development Washington, DC 20590	13. Type of Report and Period Covered Final Report December 1982 - July 1986
	14. Sponsoring Agency Code NRD-13

15. Supplementary Notes
*Department of Aeronautics & Astronautics
Air Force Institute of Technology
Wright-Patterson Air Force Base, Ohio 45433

16. Abstract

Performance requirements are developed which define the kinematic and kinetic response of the head for a seated subject exposed to frontal, lateral or oblique impact. Response is expressed in terms of variables which are readily measured in an anthropomorphic dummy and which are useful in injury prediction. The performance requirements are based on volunteer tests conducted by the U.S. Department of Navy, Naval Biodynamics Laboratory (NBDL) in which a four-point restraint system and a singular type of impact profile are employed. Other NBDL volunteer tests and volunteer and cadaver tests conducted by Wayne State University are used to evaluate the effects of variation in impact profile, type of restraint system and level of muscle activity.

17. Key Words	18. Distribution Statement
Automotive Safety, Biomechanics, Human Impact Response, Anthropomorphic Dummies	DOCUMENT IS AVAILABLE TO THE PUBLIC THROUGH THE NATIONAL TECHNICAL INFORMATION SERVICE, SPRINGFIELD, VIRGINIA 22161

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	160	

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

PREFACE

It was necessary to analyze a large amount of human response data in order to establish quantitative measures that evaluate fidelity of head response in an anthropomorphic dummy. To facilitate this work, software was developed to automate the data processing on the VAX computer of the National Highway Traffic Safety Administration. This volume is a guide to the use of that software.

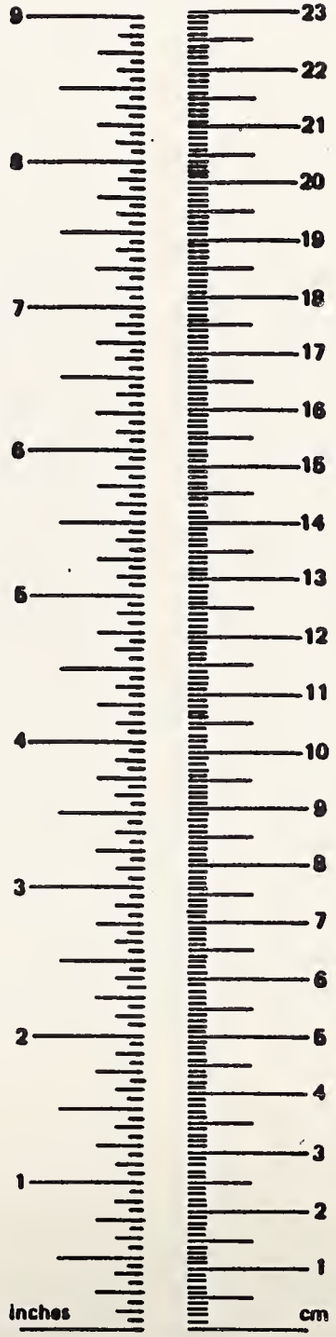
The software was developed under the direction of Dr. C. H. Spenny formerly of the Transportation Systems Center (TSC), by Messrs. J. Burstein, D. A. Gordon, T. Peters and R. Stevens of the Systems Development Corporation, an on-site ADP contractor at TSC.

METRIC CONVERSION FACTORS

Approximate Conversions to Metric Measures

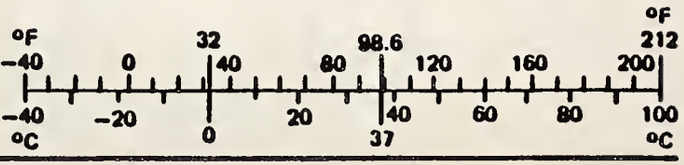
Symbol	When You Know	Multiply by	To Find	Symbol
LENGTH				
in	inches	*2.5	centimeters	cm
ft	feet	30	centimeters	cm
yd	yards	0.9	meters	m
mi	miles	1.6	kilometers	km
AREA				
in^2	square inches	6.5	square centimeters	cm^2
ft^2	square feet	0.09	square meters	m^2
yd^2	square yards	0.8	square meters	m^2
mi^2	square miles	2.6	square kilometers	km^2
	acres	0.4	hectares	ha
MASS (weight)				
oz	ounces	28	grams	g
lb	pounds	0.45	kilograms	kg
	short tons (2000 lb)	0.9	tonnes	t
VOLUME				
tsp	teaspoons	5	milliliters	ml
Tbsp	tablespoons	15	milliliters	ml
fl oz	fluid ounces	30	milliliters	ml
c	cups	0.24	liters	l
pt	pints	0.47	liters	l
qt	quarts	0.95	liters	l
gal	gallons	3.8	liters	l
ft^3	cubic feet	0.03	cubic meters	m^3
yd^3	cubic yards	0.76	cubic meters	m^3
TEMPERATURE (exact)				
°F	Fahrenheit temperature	5/9 (after subtracting 32)	Celsius temperature	°C

*1 in. = 2.54 cm (exactly). For other exact conversions and more detail tables see NBS Misc. Publ. 286, Units of Weight and Measures. Price $2.25 SD Catalog No. C13 10 286.



Approximate Conversions from Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
LENGTH				
mm	millimeters	0.04	inches	in
cm	centimeters	0.4	inches	in
m	meters	3.3	feet	ft
m	meters	1.1	yards	yd
km	kilometers	0.6	miles	mi
AREA				
cm^2	square centimeters	0.16	square inches	in^2
m^2	square meters	1.2	square yards	yd^2
km^2	square kilometers	0.4	square miles	mi^2
ha	hectares (10,000 m^2)	2.5	acres	
MASS (weight)				
g	grams	0.035	ounces	oz
kg	kilograms	2.2	pounds	lb
t	tonnes (1000 kg)	1.1	short tons	
VOLUME				
ml	milliliters	0.03	fluid ounces	fl oz
l	liters	2.1	pints	pt
l	liters	1.06	quarts	qt
l	liters	0.26	gallons	gal
m^3	cubic meters	36	cubic feet	ft^3
m^3	cubic meters	1.3	cubic yards	yd^3
TEMPERATURE (exact)				
°C	Celsius temperature	9/5 (then add 32)	Fahrenheit temperature	°F



CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The data retrieval, analysis and display software described in this volume consists of a general purpose data manipulation program, the Data Retrieval and Display (DRD) program, and a pair of specialized analysis programs, HEAD and NECK.

The DRD program is user friendly and is designed to quickly and efficiently retrieve and graphically display data on head and neck response. As currently programmed it can be used with the test data from the Naval Biodynamics Laboratory, Wayne State University (WSU), and the University of Michigan Transportation Research Institute (UMTRI) that is tabulated in this volume. This data base consists of 380 tests.

HEAD and NECK are specialized programs written for use at the Transportation Systems Center in analysis of head and neck response. The response variables that are calculated by these programs are integrated with the test data and displayed using the DRD program.

Figure 1-1 is a block diagram representation of the software. All software is written in Fortran for operation on a VAX/VMS computer. The Fortran coding is included as Appendix C of this volume.

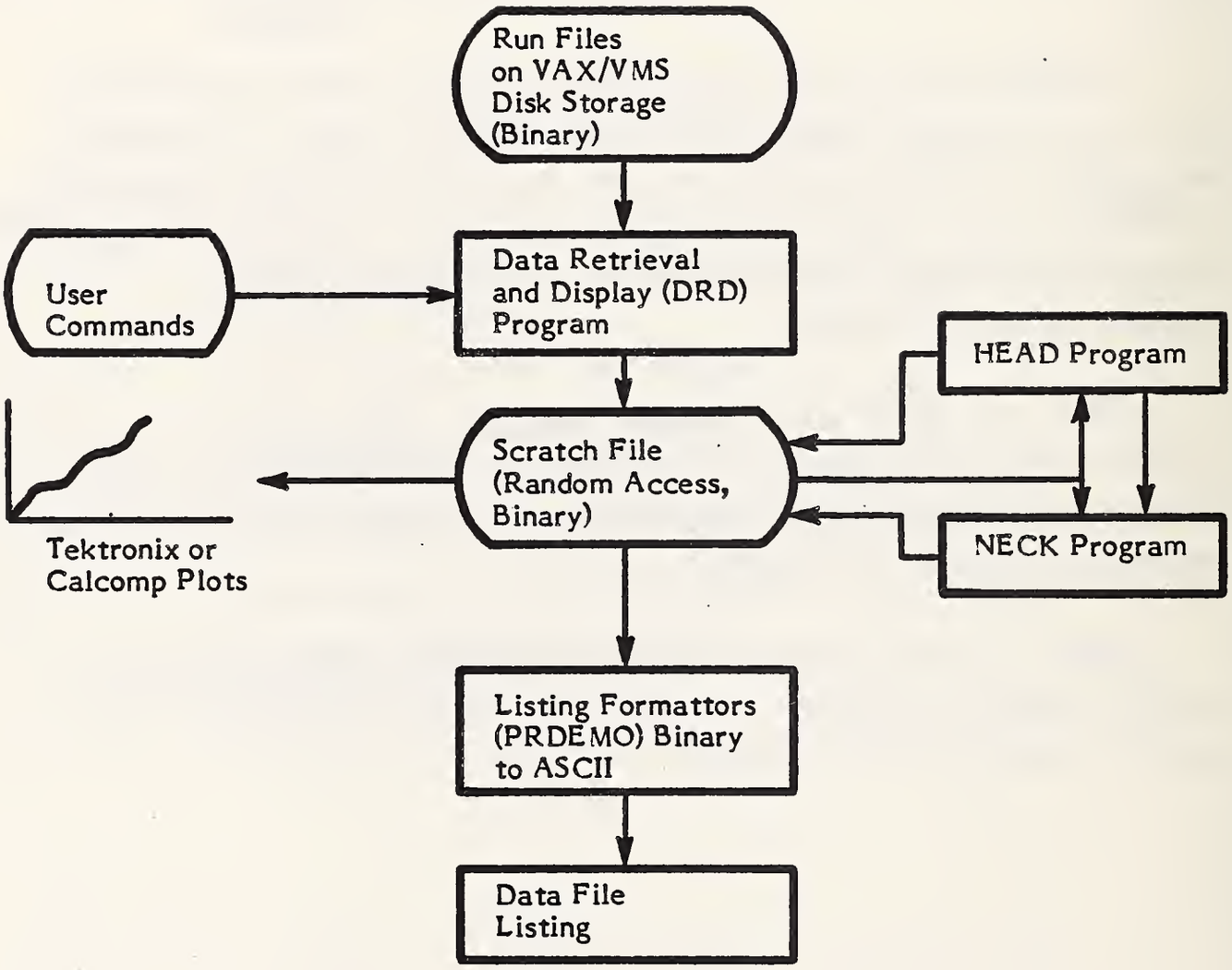


FIGURE 1-1. BLOCK DIAGRAM REPRESENTATION OF THE DATA RETRIEVAL, ANALYSIS AND DISPLAY SOFTWARE

2. PROGRAMS FOR DATA RETRIEVAL AND DISPLAY (DRD)

2.1 DESCRIPTION

The user may retrieve any of the variables (records) from the run files that reside in disk storage. These variables are selected by specifying the run number and variable name for each variable requested. The variables that may be retrieved are those listed in Appendix A of this volume. The total number of test variables is 92.

To reduce disk storage requirements, the run files are written in binary format and each run includes only 66 of the test variables. The data that is not present consists of; (1) linear velocity data derived from photography, and (2) linear velocity and displacement data as derived from sensor (accelerometer) measurements. When the user requests one of the variables not present, it is reconstructed by differentiating photographic displacement or integrating sensor acceleration, as appropriate. This operation is transparent to the user and reproduces the test results to six significant figures.

Each run file consists of a series of records, the first being a header record which contains the run number, subject number and other parameters describing the overall test. The header record is followed by the 66 records of stored variables. Each sensor (accelerometer) record has the same format, consisting of 598 fields of test data. All NBDL sensor measurements were taken at .0005 sec timesteps, thus allowing the storage of response data for just under 0.3 sec. Sensor data is not available for the Wayne State and UMTRI data.

Each photographic record within a run is identical in format and size. However, the number of fields varies from run to run and is determined by the number of frames digitized. In addition, the time step varies between runs as determined by camera speed. One of the photographic records is the variable, TIME, whose field entries are the times of occurrence of film frames which were digitized.

Data records requested by the user, up to a maximum of 100, are extracted into a scratch file. The scratch file is binary and becomes the source file for use by other portions of the DRD program and by HEAD and NECK. DRD capabilities,

in addition to data extraction, include: (1) filing and retrieving scratch files, (2) creation of new variables by performing basic mathematical operations on one or more variables in the scratch file, and (3) graphic display. Use of the DRD capabilities is described in Section 2.2 which follows.

It should be noted that the graphic display module recognizes sensor and photographic variables and correctly plots them versus time, even on the same graph. However, there are no program checks to prevent the creation of illogical new variables. In particular, the user is cautioned against combining (and cross plotting) photographic and sensor variables.

2.2 USE OF THE DRD PROGRAM

To use the DRD program issue the VAX/VMS DCL command;

```
$ RUN TSCPROG.SRC.PROGS.NECK.ANALYSIS XTRAC
```

The program will issue a prompt (XTR>) and expect the user to type in one of the following commands;

CLEAR	END	NORMALIZE
EXTRACT	ADD	STANDEV
DIRECTORY	SUBTRACT	DSPLAY
FILE	VMAGNITUDE	
GET	DIVIDE	

The first three characters of each command must be entered. The remaining characters are optional. The following sections describe the syntax of each of these commands.

2.2.1 CLEAR Command

This command sets the number of entries in the directory to zero. The scratch file is thus cleared and contains no data records. This command should be used before a new scratch file is made up, or to clear the file in case errors are made during data extraction.

The use of the optional 'n' with this command sets the number of entries in the directory to the integer value 'n', allowing the user better control over the extracted data file. The default value when 'n' is not specified is 100. The user can extract variables, produce new variables (seen in the following sections) and overwrite these variables if results are unsatisfactory.

```
XTR> CLEAR      or
XTR> CLEAR 20
```

2.2.2 EXTRACT Command

This command is used to extract records from the original binary format files. These records are then written to a random access file named SCRTCH.DAT. Individual tests are identified by run number and individual variables are identified by variable name. The keyword 'ALL' may be used in place of a variable name to extract all variables of any given run. Run numbers corresponding to the tests of Volume I, Appendix A, are repeated in Table 2-1 for convenience. A listing of variables is given in Appendix A of this volume.

```
XTR> EXTRACT RUN LX1916 VAR VNXSOP or
XTR> EXTRACT RUN LX1916 ALL
```

Up to 100 variables may be extracted and may reside in the scratch file at any one time. Note that extracting 'ALL' variables places 92 variables in the scratch file, leaving room for only eight more.

2.2.3 DIRECTORY Command

This command causes a listing of the scratch file to be output to the user's terminal. This listing shows which records are contained in the scratch file. The listing for each record includes the variable numbers, variable name, run number,

TABLE 2-1. LISTING OF TEST RUN NUMBERS RECOGNIZED BY THE DRD PROGRAM

NBDL Frontal Test Runs

LX3524	LX3525	LX3530	LX3531	LX3536	LX3537
LX3544	LX3548	LX3550	LX3558	LX3573	LX3578
LX3583	LX3616	LX3779	LX3780	LX3782	LX3783
LX3785	LX3786	LX3788	LX3789	LX3791	LX3793
LX3794	LX3796	LX3797	LX3798	LX3800	LX3801
LX3803	LX3804	LX3805	LX3807	LX3808	LX3809
LX3812	LX3814	LX3815	LX3817	LX3818	LX3819
LX3821	LX3822	LX3823	LX3824	LX3833	LX3837
LX3839	LX3840	LX3841	LX3842	LX3851	LX3852
LX3854	LX3856	LX3857	LX3858	LX3869	LX3870
LX3871	LX3872	LX3875	LX3876	LX3878	LX3880
LX3882	LX3883	LX3885	LX3886	LX3887	LX3889
LX3890	LX3893	LX3894	LX3895	LX3898	LX3900
LX3901	LX3903	LX3904	LX3906	LX3908	LX3909
LX3913	LX3914	LX3916	LX3918	LX3920	LX3921
LX3924	LX3926	LX3927	LX3928	LX3939	LX3940
LX3941	LX3942	LX3944	LX3945	LX3946	LX3948
LX3949	LX3950	LX3951	LX3953	LX3954	LX3955
LX3957	LX3958	LX3959	LX3961	LX3962	LX3963
LX3965	LX3968	LX3969	LX3970	LX3972	LX3982
LX3983	LX3985	LX3986	LX3987	LX3989	LX3990
LX3991	LX3993	LX3994	LX3995	LX3997	LX3998
LX3999					

NBDL Lateral Test Runs

LX1454	LX1456	LX1457	LX1458	LX1468	LX1470
LX1471	LX1474	LX1475	LX1484	LX1487	LX1501
LX1503	LX1504	LX1505	LX1507	LX1509	LX1510
LX1512	LX1513	LX1524	LX1525	LX1526	LX1528
LX1785	LX1793	LX1831	LX1860	LX1874	LX1916
LX1960	LX1998	LX2010	LX2013	LX2027	LX2032
LX2056	LX2060	LX2072	LX2090	LX2102	LX2124
LX2137	LX2148	LX2151	LX2182	LX2282	LX2294
LX2298	LX2302	LX2313	LX2326	LX2338	LX2341
LX2355	LX4050	LX4052	LX4053	LX4054	LX4055
LX4057	LX4058	LX4059	XL4060	LX4068	LX4069
LX4070	LX4071	LX4073	LX4074	LX4075	LX4076
LX4078	LX4079	LX4080	LX4081	LX4083	LX4084
LX4085	LX4088	LX4089	LX4090	LX4092	LX4093
LX4094	LX4095	LX4097	LX4098	LX4099	LX4100
LX4104	LX4107	LX4109	LX4110	LX4111	LX4112
LX4114	LX4115	LX4116	LX4118	LX4119	LX4120
LX4123	LX4124	LX4125	LX4126	LX4128	LX4129
LX4130	LX4131	LX4133	LX4134	LX4135	LX4137
LX4138	LX4139	LX4140	LX4142	LX4143	LX4144
LX4145	LX4147	LX4148	LX4149	LX4151	LX4153
LX4155					

NBDL Oblique Test Runs

LX2763	LX2770	LX2772	LX2784	LX2786	LX2799
LX2801	LX2813	LX2815	LX2827	LX2829	LX2843
LX2872	LX2876	LX2916	LX2955	LX2973	LX2979
LX2982	LX2985	LX2988	LX3049	LX3053	LX3061
LX3065	LX3077	LX3085	LX3089	LX3093	LX3097
LX3100	LX3102	LX3106	LX3122	LX3129	LX3133
LX3145	LX3148	LX3153	LX3158	LX3417	LX4159
LX4161	LX4162	LX4163	LX4164	LX4166	LX4167
LX4168	LX4170	LX4171	LX4172	LX4234	LX4235
LX4236	LX4237	LX4238	LX4240	LX4241	LX4242
LX4243	LX4244	LX4246	LX4247	LX4248	LX4249
LX4251	LX4259	LX4260	LX4261	LX4263	LX4264
LX4265	LX4266	LX4268	LX4269	LX4270	LX4271
LX4276	LX4277	LX4280	LX4281	LX4282	LX4284
LX4286	LX4287	LX4288	LX4290	LX4291	LX4292
LX4293	LX4295	LX4296	LX4297	LX4298	LX4301
LX4302	LX4303	LX4305	LX4306	LX4307	LX4309
LX4310	LX4313	LX4314	LX4316		

WSU Frontal Test Runs

DOT307	DOT308	DOT309	DOT310	DOT314	DOT331
DOT332	DOT333	DOT343	DOT345	DOT453	DOT454
DOT455					

UMTRI Frontal Test Runs

T76008

minimum value, maximum value, and number of data points for that record. This listing contains only those records which are part of the scratch file at the time the DIRECTORY command is given. If the scratch file contains no records, the message "EMPTY" is sent to the terminal and the user is prompted for the next command.

2.2.4 FILE Command

This command is used to place the contents of the scratch file in a permanent disk file specified by the given filename. The file is written in scratch file format, making it possible to store and reuse the data at another time.

If the permanent disk file does not exist, it is created, the contents of the scratch file are placed in it, and the scratch file is deleted.

If the permanent disk file does exist, the contents of it are written to a file with the same name and a version number one higher than that which already exists. The filename must be specified in single quotes using VAX/VMS filenaming convention.

```
XTR> FILE  'NEWDAT.DAT'  results in the prompt;
Variable #'s or ALL   1,3,18,25  or
Variable #'s or ALL   ALL
```

2.2.5 GET Command

The GET command gets files that were filed using the FILE command. Filenames must include the user default directory and must be enclosed in single quotes. The file specified must be in scratch file format e.g, if the NEWDAT.DAT file in the above example is created in directory [CURTS.TEST] the GET command looks like:

```
XTR> GET '[CURTS.TEST] NEWDAT.DAT'
```

2.2.6 END Command

The END command ends operation of the program. Control is returned to DCL (Digital Command Level) of the VMS operating system.

```
XTR> END
```

2.2.7 ADD Command

This command adds the values of the fields of the data record specified by the first variable name to those of the second variable name and sets the results in a new record which is given the name specified by the third variable name parameter. This name must be enclosed in single quotes. The data records used to produce the new variable must already reside in the scratch file. The new variable is then written out as a new record on the scratch file.

```
XTR> ADD RUN LX1960 VAR AAXXOS AAYXOS 'XPLUSY'
```

2.2.8 SUBTRACT Command

This is the same as the ADD command except that the values of the record selected by the second variable name are instead subtracted from the record specified by the first variable name.

```
XTR> SUBTRACT RUN LX1960 VAR AAXXOS AAYXOS 'XMNUSY'
```

2.2.9 VMAGNITUDE Command

This command computes the vector magnitude of a vector specified by the first three variable name fields following the VARIABLE command. These fields specify the names of the x, y, and z components of a three dimensional vector. These variables must already be contained in the scratch file. They are used to compute the vector magnitude, v, according to:

$$V = \quad x^2 + y^2 + x^2$$

where the magnitude "v" is specified by the fourth variable name parameter. (Note that this parameter is a new variable name which is not one of the names on the list in Table 2-1. Also, since this name may contain from one to six characters, it is enclosed by single quotes in order to aid the extraction module in its syntax analysis.) A vector magnitude is computed and stored in the scratch file.

XTR> VMAGNITUDE RUN LX1960 VAR AAXXOS AAYXOS AAZXOS 'AAVMAG'

2.2.10 DIVIDE Command

This command divides by a constant the values of the fields of the data record specified by the first variable name to produce a new record in the scratch file, which is specified by the second variable name. The new name must be enclosed in single quotes. The data records for the first variable must already reside in the scratch file.

XTR> DIV RUN LX1960 VAR AAXOS -.396 'SCALED'

2.2.11 CONSTANT Command

This command adds a constant to the values of the fields of the data record specified by the first variable name to produce a new record in the scratch file, which is specified by the second variable name. The new name must be enclosed in single quotes. The data records for the first variable must already reside in the scratch file.

XTR> CON RUN LX1960 VAR AAXXOS +.200 'BIAS'

2.2.12 NORMALIZE Command

This command divides the values of each field of the data record specified by the first variable name by the value in the first field to produce a new record in the scratch field, which is specified by a second variable name. The new name must be enclosed in single quotes. The data records for the first variable name must already reside in the scratch file.

XTR> NOR RUN LX1960 VAR AAXOS 'NORAAX'

2.2.13 STANDEV Command

The STANDEV command calculates mean, mean plus standard deviation, and mean minus standard deviation and standard deviation at each timestep for a given variable. The scratch file is searched for each occurance of the specified variable and each occurance is used in the calculation of mean and standard deviation at each timestep. The three new variables are named according to the following convention;

MOXXXX
MPXXXX
MMXXXX
SDXXXX

where XXXX stands for the first three letters of the given variable plus either a P or an S, depending on whether it is a photo or sensor variable. MO stands for mean, MP stands for mean plus standard deviation, MM stands for mean minus standard deviation and SD stands for standard deviation

```
XTR> STANDEV VAXXOS
```

This command would create the four new variables MOVAXS, MPVAXS, MMVAXS and SDVAXS.

2.2.14 DSPLAY Command

The DSPLAY command places the user in the display module of the DRD program. This module issues a different prompt, 'DSP '. Once in this module, the user chooses from the following commands;

PLOT	DIRECTORY	DSPLAY
XTRAC	END	COPY

A PLOT command is entered for each curve to be plotted on a display page. Multiple plots may be selected for the same display. Up to 20 PLOT commands may be entered. When all of the plots have been requested a DSPLAY command is entered and the display is output to the graphics terminal. Entering a COPY command rather than a DSPLAY command causes output to be displayed on the graphics screen and then generates a copy of a hardcopy unit.

2.2.14.1 PLOT Command - Each plot command selects two variables to be used as the x and y values of a curve to be plotted two dimensionally on a graphics device such as a Tektronix 4010, 4114, or 4115. The first variable name is used as the x coordinate, the second variable name as the y coordinate. All variables selected must already be in the scratch file before plot commands are entered. Time is (of course) a valid variable name for both photographic and sensor data. When plotting sensor variables, time is calculated when necessary. When plotting photographic variables, the time variable must already be in the scratch file.

The user may choose from several line formats. Solid lines, solid lines with markers, or dashed lines are the available options. Marker value is an optional argument for this command, it must be integer and may be between -5 and 8. Markers come in eight flavors (for example: squares, circles, or triangles) and are denoted by the integers 1 through 8. Dashed lines come in five types, denoted by the integers -5 through -1. If no marker value is specified, the default is 0, the solid line.

The user may also optionally specify the number of points to be plotted. This is useful when plotting photographic data vs. sensor data when the number of points varies or when the user is interested in viewing a smaller segment of data. The marker argument must be included (it may be a 0) when using the points feature. Otherwise, the point value will be interpreted as a marker value.

DSP > PLOT RUN LX1916 TIME AAXXOS	will give a plot of time vs. aaxxos with a solid line (the default).
DSP > PLOT RUN LX3958 TIME ANXXOS 1	will give a plot with circles used as markers.

DSP > PLOT RUN LX3958 TIME VNXXOS 1 250 will give the first 250 points of this plot with circles as markers.

DSP > PLOT RUN LX3958 TIME VNXSOP -2 will produce the plot with a dashed line (no markers).

2.2.14.2 DSPLAY Command - The DSPLAY command is used to mark the end of the plot specifications and causes generation of the plot on the current graphics device. All plot commands issued since the most recent other DSPLAY command will be plotted on a single graph. The axes will be scaled to accommodate all variables being plotted. They will be labelled with the dimensions of the first pair of variables to be plotted. Figure 2-1 shows the multiple plots resulting from the sample plot commands in section 2.2.14.1.

DSP > DSPLAY

2.2.14.3 COPY Command - The COPY command also marks the end of plot specifications and causes generation of the plot on the graphics screen. Additionally, COPY causes a hard-copy device (a Tektronix printer) to copy the screen when plotting has finished.

DSP > COPY

2.2.14.4 DIRECTORY Command - The DIRECTORY command is the same as was described in 2.2.3.

2.2.14.5 XTRAC Command - The XTRAC command returns the user to the extraction module when the current plotting has been finished so that additional data files may be processed.

DSP > XTRAC

2.2.14.6 END Command - The directory command is the same as was described in 2.2.6.

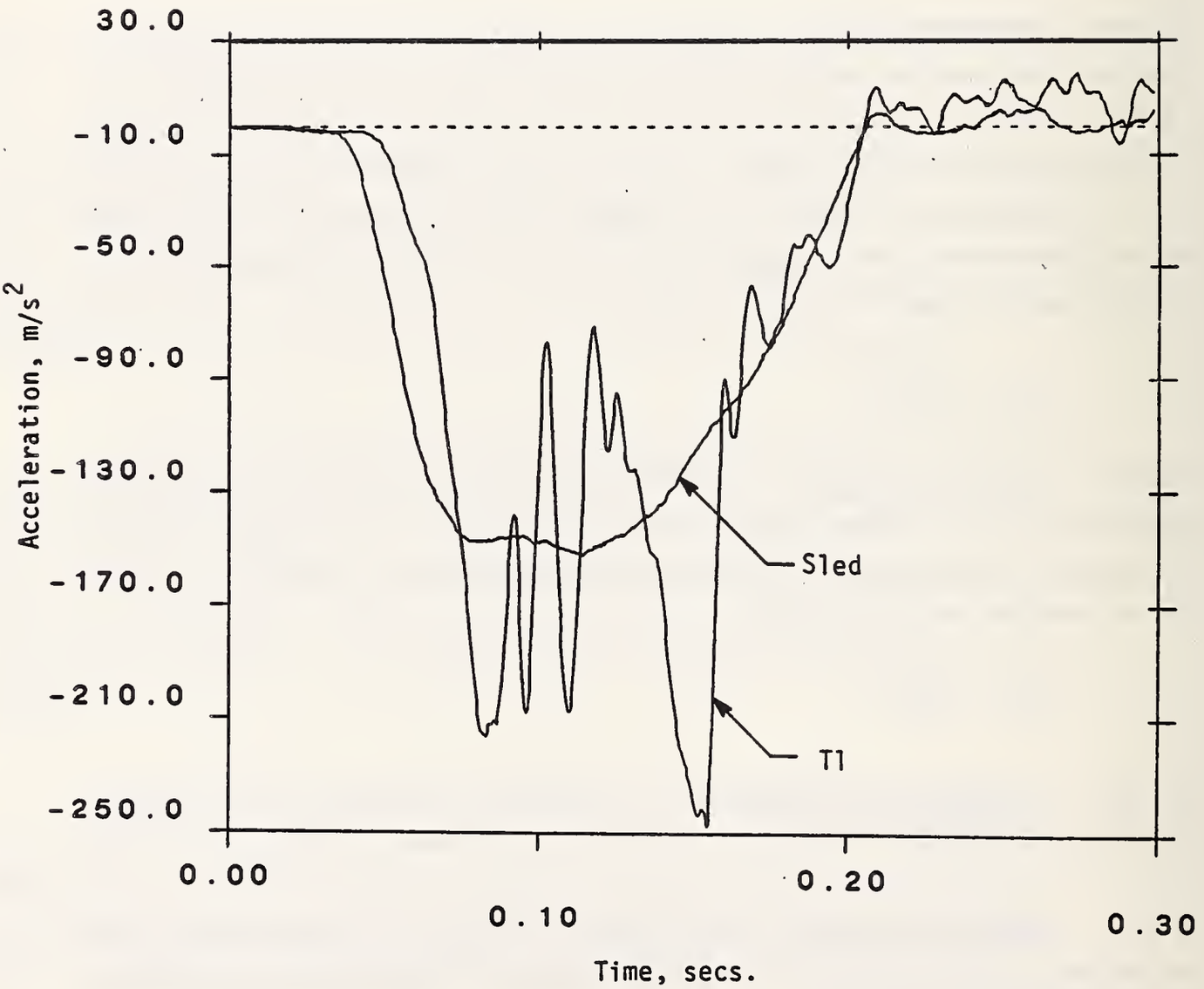


FIGURE 2-1. EXAMPLE OF MULTICURVE PLOTTING CAPABILITY

3. PROGRAM FOR CALCULATION OF HEAD KINEMATIC AND LOAD RESPONSE (HEAD)

3.1 DESCRIPTION

The kinematic and load response of the head, as described in Volume I by equations (2), (3), (5), (6), (24), (25) and (26) are calculated by the HEAD program. This program is designed for use in conjunction with data retrieved by the DRD. Table 3-1 lists the NBDL defined variables which HEAD reads from a DRD scratch file.(1)

The input variables are a combination of photographic and accelerometer derived variables as indicated by the final letter of the Fortran name-P for photo and S for accelerometer (sensor). Calculations are made by HEAD at the times at which photo data is digitized. A subroutine within HEAD interpolates sensor data, using the two nearest points in time to produce acceleration and velocity data at the photo time points.

Subject specific data as indicated in Table 3-2 is coded into HEAD and called as required to match the test subject whose data has been read from the scratch file.

Table 3-3 lists the output variables which are calculated and attached to the DRD scratch file which ran HEAD. Note six variables are also stored in a separate file for use by NECK, indicating that the latter program can be run only in conjunction with HEAD. The block diagram of Figure 1-1 indicates how HEAD interfaces with the DRD package and with NECK.

3.2 USE OF THE HEAD PROGRAM

The variables of Table 3-1 must be present in the scratch file of the DRD. When a variable name appears more than once in the scratch file, the first is selected for use by HEAD.

---

(1) Neck variables ANXXOS and VNXXOS are included as input to HEAD in order to enter them in the database at the photo timesteps. They are not required for head computations.

TABLE 3-1. INPUT VARIABLES FOR THE HEAD PROGRAM

FORTRAN SYMBOL	ANALYSIS SYMBOL (VOLUME I)	GENERAL DEFINITION[1]
AAXXOS, AAYXOS, AAZXOS	a_{Ax}, a_{Ay}, a_{Az}	Linear acceleration of the head anatomical origin
ANXXOS	——	Linear acceleration of the T1 anatomical origin
PHAOXP, PHBO2P, PHCO3P	$\theta_{Hx}, \theta_{Hy}, \theta_{Hz}$	Euler angle description of head rotation
PNAOXP, PNBO2P, PNCO3P	$\theta_{Nx}, \theta_{Ny}, \theta_{Nz}$	Euler angle description of T1 vertebral rotation
QHAOXS, QHBOXS, QHCOXS	$\alpha_x, \alpha_y, \alpha_z$	Angular acceleration of the head
RHAOXS, RHBOXS, RHOCOXS	$\omega_x, \omega_y, \omega_z$	Angular velocity of the head
TIME	——	Time at which photo data is digitized
VNXXOS	——	Linear velocity of the T1 anatomical origin

(1) The input variables are fully defined in Appendix A.

TABLE 3-2. SUBJECT SPECIFIC DATA STORED WITHIN THE HEAD PROGRAM

FORTRAN SYMBOL	ANALYSIS SYMBOL (VOLUME I)	GENERALIZED DESCRIPTION
IX, IY, IZ	I_{xx}, I_{yy}, I_{zz}	Centroidal mass moment of inertia coefficients of the instrumented head about the X, Y, and Z axis, respectively, of the head anatomical coordinate system
MH	M_H	Mass of the instrumented head
PXY, PXZ PYX, PYZ PZX, PZY	I_{xy}, I_{xz}, I_{Yz}	Centroidal mass product of inertia coefficients of the instrumented head
RGOX, RGOY, RGOZ	$r_{G/Ox}$, $r_{G/Oy}$, $r_{G/Oz}$	Position of the head center-of-gravity relative to the occipital condylar point (head anatomical components).

TABLE 3-3. OUTPUT VARIABLES FOR THE HEAD PROGRAM

FORTRAN SYMBOL	ANALYSIS SYMBOL (VOLUME I)	GENERALIZED DEFINITION[1]
AGXP, AGYP, AGZP	a_{Gx}, a_{Gy}, a_{Gz}	Linear acceleration of the head center-of-gravity
ANXOP	——	Linear acceleration of the T1 anatomical origin
FOXLP, FOYLP, FOZLP/ TOXLP, TOZLP, TOZLP	——	Laboratory coordinate system components of force/torque applied to the head by the neck[2]
FOXP, FOYP, FOZP	F_{Ox}, F_{Oy}, F_{Oz}	Head anatomical coordinate system components of force/torque applied to the head by the neck
TOXP, TOYP, TOZP	T_{Ox}, T_{Oy}, T_{Oz}	
FOXTP, FOYTP, FOZTP TOXTP, TOYTP, TOZTP	——	To anatomical coordinate system components of force/torque applied to the head by the neck
THTIXP, THTYP	ϕ_x, ϕ_y	Angular orientation of the head relative to the torso.

(1) The output variables are fully defined in Appendix B.

(2) Variables which are stored for use by program NECK.

4. PROGRAM FOR CALCULATION OF NECK KINEMATIC AND LOAD RESPONSE (NECK)

4.1 DESCRIPTION

The kinematic and load response of the neck, as described in Volume I by equations (1), (2), (3), (9), (11), (14), (22) and (23) are calculated by the NECK program. The program is designed for use in conjunction with the DRD and HEAD programs as noted in the previous section. Table 4-1 lists the NBDL defined variables which NECK reads from a DRD scratch file and from HEAD.* All input variables are photo-derived so there is no need for interpolation of sensor data as in HEAD.

Subject specific data as indicated in Table 4-2 is coded into NECK and is called as required to match the test subject whose data has been read from the scratch file. Parameters ARP, BR and DNZMN are used to correct the data for observed error in the vertical mounted position of the T1 instrumentation as described in Section 5.1. This data correction is made in NECK thereby preserving the original database for use by other researchers. Provision has not been made for display of the corrected variable, DNZSOP, since it was not required for presentation of study results.

Table 4-3 lists the variables which are calculated and attached to the DRD scratch file which was used to run NECK. Figure 1-1 is a block diagram which indicates how NECK interfaces with DRD and HEAD.

4.2. USE OF THE NECK PROGRAM

The variable of Table 4-1 must be present in the scratch file of the DRD. When a variable name appears more than once in the scratch file, the first is selected for use by NECK.

---

*Reduced versions of HEAD and NECK, that require only photo-derived variables were used to process data from Wayne State University and the University of Michigan. In the reduced programs, there is no input to NECK required from HEAD.

TABLE 4-1. INPUT VARIABLES FOR THE NECK PROGRAM

FORTRAN SYMBOL	ANALYSIS SYMBOL (VOLUME I)	GENERALIZED DEFINITION[1]
FROM DRD SCRATCH FILE:		
DAXSOP, DAYSOP, DAZSOP	r_A	Displacement of the head relative to the sled
DNXSOP, DNYSOP, DNZSOP	r_T	Displacement of the T1 vertebral body relative to the sled
PHAOXP, PHBO2P, PHCO3P	θ_{Hx}, θ_{Hy}, θ_{Hz}	Euler angle description of head rotation
PNAOXP, PNBO2P, PNCO3P	θ_{Nx}, θ_{Ny}, θ_{Nz}	Euler angle description of T1 vertebral rotation
TIME	—	Time at which photo data is digitized
FROM HEAD PROGRAM:		
FOXLP, FOYLP, FOZLP TOXLP, TOYLP, TOZLP applied to the head by the neck.	—	Laboratory coordinate system components of force/torque

(1) The input variables are fully defined in Appendix A.

TABLE 4-2. SUBJECT SPECIFIC DATA STORED WITHIN THE NECK PROGRAM

FORTRAN SYMBOL	ANALYSIS SYMBOL (VOLUME I)	GENERALIZED DEFINITION
ARP, BR, DNZMN	$a^*, \frac{\mu 11}{(\sigma_R)^2}$, (DNZSOP)M	a, b, r_{T_Z} Least squares fit parameters for correcting T1 vertical position DNZSOP
RGAX, RGAZ	$r_{G/Ax}, r_{G/Az}$	Position of the head center-of-gravity relative to the head anatomical origin (head anatomical components ·

$$*a = \overline{(DNZSOP)}_M + \frac{\mu 11}{(\sigma_R)^2} (RATIP)_M$$

TABLE 4-3. OUTPUT VARIABLES FOR THE NECK PROGRAM

FORTRAN SYMBOL	ANALYSIS SYMBOL (VOLUME I)	GENERALIZED DEFINITIONS[1]
PSI	ψ_C	Externally observed head twist relative to the torso
RATIP	$r_{O/T}-r_{O/A}$	Distance from the T1 vertebral to the head anatomical origin
ROTIP	$r_{O/T}-r_{O/C}$	Distance from the T1 vertebral to the occipital condylar point ("Neck chord length")
TENTXP, TENTYP	θ_x, θ_y	Angular orientation of the neck chord vector relative to the torso
TETHNP	ψ_I	Internally measured head twist about the head z-axis
T1XLP, T1YLP, T1ZLP	components of the torque	Laboratory coordinate system applied to the neck by the torso
T1XTP, T1YTP T1ZTP	——	To coordinate system components of the torque applied to the neck by the torso

(1) The output variables are fully defined in Appendix B.

APPENDIX A

DEFINITION OF VARIABLES CONTAINED IN THE NBDL DATBASE

(REPRODUCED FROM REF. 4)

AAXXOS The component of linear acceleration of the head anatomical origin along the X axis of the laboratory coordinate system with respect to the fixed laboratory coordinate system as derived from mouth mount accelerometer data.

AAAYXOS The component of linear acceleration of the head anatomical origin along the Y axis of the laboratory coordinate system with respect to the fixed laboratory coordinate system as derived from mouth mount accelerometer data.

AAZXOS The component of linear acceleration of the head anatomical origin along the Z axis of the laboratory coordinate system with respect to the fixed laboratory coordinate system as derived from mouth mount accelerometer data.

QHAOXS Angular acceleration of the head about the X axis of head anatomical coordinate system as derived from mouth mount accelerometer data.

QHBOXS Angular acceleration of the head about the Y axis of the head anatomical coordinate system as derived from mouth mount accelerometer data.

QHCOXS Angular acceleration of the head about the Z axis of the head anatomical coordinate system as derived from mouth mount accelerometer data.

VAXXOS The component of linear velocity of the head anatomical origin along the X axis of the laboratory coordinate system with respect to the fixed laboratory coordinate system as derived from mouth mount accelerometer data.

VAXSOS The component of linear velocity of the head anatomical origin along the X axis of the sled coordinate system with respect to the sled coordinate system as derived from accelerometer data. The sled coordinate system is aligned with the laboratory coordinate system and translates with the sled.

VAYXOS The component of linear velocity of the head anatomical origin along the Y axis of the laboratory coordinate system with respect to the fixed laboratory coordinate system as derived from mouth mount accelerometer data.

VAYSOS The component of linear velocity of the head anatomical origin along the Y axis of the sled coordinate system with respect to the sled coordinate system as derived from accelerometer data. The sled coordinate system is aligned with the laboratory coordinate system and translates with the sled.

VAZXOS The component of linear velocity of the head anatomical origin along the Z axis of the laboratory coordinate system with respect to fixed laboratory coordinate system as derived from mouth mount accelerometer data.

VAZSOS The component of linear velocity of the head anatomical origin along the Z axis of the sled coordinate system with respect to the sled coordinate system as derived from accelerometer data. The sled coordinate system is aligned with the laboratory coordinate system and translates with the sled.

RHAOXS Angular velocity of the head about the X axis of the head anatomical coordinate system as derived from mouth mount accelerometer data.

RHBOXS Angular velocity of the head about the Y axis of the head anatomical coordinate system as derived from mouth mount accelerometer data.

RHCOXS Angular velocity of the head about the Z axis of the anatomical coordinate system as derived from mouth mount accelerometer data.

DAXXOS The component of linear displacement of the head anatomical origin along the X axis of the laboratory coordinate system with respect to the fixed laboratory coordinate system as derived from mouth mount accelerometer data.

DAXSOS The component of linear displacement of the head anatomical origin along the X axis of the sled coordinate system with respect to the sled coordinate system as derived from accelerometer data. The sled coordinate system is aligned with the laboratory coordinate system and translates with the sled.

DAYXOS The component of linear displacement of the head anatomical origin along the Y axis of the laboratory coordinate system with respect to the fixed laboratory coordinate system as derived from mouth mount accelerometer data.

DAYSOS The component of linear displacement of the head anatomical origin along the Y axis of the sled coordinate system with respect to the sled coordinate system as derived from accelerometer data. The sled coordinate system is aligned with the laboratory coordinate system and translates with the sled.

DAZXOS The component of linear displacement of the head anatomical origin along the Z axis of the laboratory coordinate system with respect to the fixed laboratory coordinate system as derived from mouth mount accelerometer data.

DAZSOS The component of linear displacement of the head anatomical origin along the Z axis of the sled coordinate system with respect to the sled coordinate system as derived from accelerometer data. The sled coordinate system is aligned with the laboratory coordinate system and translates with the sled.

PHAOXS — Angle of rotation of the head about the X axis of the head anatomical coordinate system as derived from mouth mount acceleromter data. Head anatomical coordinate system is initially aligned with the laboratory coordinate system.

PHBO2S — Same as PHAOXS except about the carried Y axis.

PHCO3S — Same as PHAOXS except about the carried Z axis.

EULER ANGLES

4HOO1S
4HOO2S
4HOO3S
4HOO4S — Quaternions - The four variables which define the angular orientation of the head anatomical coordinate system relative to the laboratory coordinate system as derived from mouth mount accelerometer.

ANXXOS — The component of linear acceleration of T_1 anatomical origin along the X axis of the laboratory coordinate system with respect to the fixed laboratory

ANYXOS — The component of linear acceleration of the T_1 anatomical origin along the Y axis of the laboratory coordinate system with respect to the fixed laboratory coordinate system as derived from T_1 mount accelerometer data.

ANZXOS — The component of linear acceleration of the T_1 anatomical origin along the Z axis of the laboratory coordinate system with respect to the fixed laboratory coordinate system as derived from T_1 mount accelerometer data.

QNAOXS — Angular acceleration of T_1 (first Thoracic vertebral body) about the Y axis of the T_1 anatomical coordinate system as derived from T_1 mount accelerometer data.

QNBOXS — Angular acceleration of T_1 (first Thoracic vertebral body) about the Y axis of the T_1 anatomical coordinate system as derived from T_1 mount accelerometer data.

QNCOXS — Angular acceleration of T_1 (first Thoracic vertebral body) about the Z axis of the T_1 anatomical coordinate system as derived from T_1 mount accelerometer data.

VNXXOS — The component of linear velocity of the T_1 anatomical origin along the X axis of the laboratory coordinate system with respect to the fixed laboratory coordinate system as derived from T_1 mount accelerometer data.

VNXSOS — The component of linear velocity of the T_1 anatomical origin along the X axis of the sled coordinate system with respect to sled coordinate system as derived from accelerometer data. The sled coordinate system is aligned with the laboratory coordinate system and translates with the sled.

VNYXOS The component of linear velocity of the T_1 anatomical origin along the Y axis of the laboratory coordinate system with respect to the fixed laboratory coordinate system as derived from T_1 mount accelerometer data.

VNYSOS The component linear velocity of the T_1 anatomical origin along the Y axis of sled coordinate system with respect to sled coordinate system as derived from accelerometer data. The sled coordinate system is aligned with the laboratory coordinate system and translates with the sled.

VNZXOS The component of linear velocity of the T_1 anatomical origin along the X axis of the laboratory coordinate system with respect to the fixed laboratory coordinate system as derived from T_1 mount accelerometer data.

VNZSOS The component of linear velocity of the T_1 anatomical origin along the Z axis of sled coordinate system with respect to sled coordinate system as derived from accelerometer data. The sled coordinate system is aligned with the laboratory coordinate system and translates with the sled.

RNAOXS Angular velocity of T_1 (first Thoracic vertebral body) about the X axis of the T_1 anatomical coordinate system as derived from T_1 mount accelerometer data.

RNBOXS Angular velocity of T_1 (first Thoracic vertebral body) about the Y axis of the T_1 anatomical coordinate system as derived from T_1 mount accelerometer data.

RNCOXS Angular velocity of T_1 (first Thoracic vertebral body) about the Z axis of the T_1 anatomical coordinate system as derived from T_1 mount accelerometer data.

DNXXOS The component of linear displacement of the T_1 anatomical origin along the X axis of the laboratory coordinate system with respect to the fixed laboratory coordinate system as derived from T_1 mount accelerometer data.

DNXSOS The component of linear displacement of the T_1 anatomical origin along the X axis of the sled coordinate system with respect to the sled coordinate system as derived from accelerometer data. The sled coordinate system is aligned with the laboratory coordinate system and translates with the sled.

DNYXOS The component of linear displacement of the T_1 anatomical origin along the Y axis of the laboratory coordinate system with respect to the fixed laboratory coordinate system as derived from T_1 mount accelerometer data.

DNYSOS — The component of linear displacement of the T_1 anatomical origin along the Y axis of the sled coordinate system with respect to the sled coordinate system as derived from accelerometer data. The sled coordinate system is aligned with the laboratory coordinate system and translates with the sled.

DNZXOS — The component of linear displacement of the T_1 anatomical origin along the Z axis of the laboratory coordinate system with respect to the fixed laboratory coordinate system as derived from T_1 mount accelerometer data.

DNZSOS — The component of linear displacement of the T_1 anatomical origin along the Z axis of the sled coordinate system with respect to the sled coordinate system as derived from accelerometer data. The sled coordinate system is aligned with the laboratory coordinate system and translates with the sled.

PNAOXS — Angle of rotation of T_1 (first Thoracic vertebral body) about the X axis of the T_1 anatomical coordinate system as derived from T_1 mount accelerometer data. The T_1 anatomical coordinate system is initially aligned with the laboratory coordinate system.

PNBO2S — Same as PNAOXS except about the carried Y axis.

PNCO3S — Same as PNB02S except about the carried Z axis.

} EULER ANGLES

4NOO1S — Quaternions - the four variables which define the angular orientation of the T_1 anatomical coordinate system relative to the laboratory coordinate system as derived from T_1 mount accelerometer data.

4NOO2S
4NOO3S
4NOO4S

ACXXOS — Linear acceleration of the sled along the X axis of the laboratory coordinate system with respect to the fixed laboratory coordinate system as measured by sled mounted accelerometer.

VCXXOS — Linear velocity of the sled along the X axis of the laboratory coordinate system with respect to the fixed laboratory coordinate system as derived from sled mounted accelerometer data.

DCXXOS — Linear displacement of the sled along the X axis of the laboratory coordinate system with respect to the fixed laboratory coordinate system as derived from sled mounted accelerometer data.

VAXSOP — The component of linear velocity of the head anatomical origin along the X axis of the sled coordinate system with respect to the sled coordinate system as derived from mouth mount photo target data. The sled coordinate system is aligned with the laboratory coordinate system and translates with the sled.

VAYSOP — The component of linear velocity of the head anatomical origin along the Y axis of the sled coordinate system with respect to the sled coordinate system as derived from mouth mount photo target data. The sled coordinate system is aligned with the laboratory coordinate system and translates with the sled.

VAXSOP — The component of linear velocity of the head anatomical origin along the Z axis of the sled coordinate system with respect to the sled coordinate system as derived from mouth mount photo target data. The sled coordinate system is aligned with the laboratory coordinate system and translates with the sled.

RHAOXP — Angular velocity of the head about the X axis of the head anatomical coordinate system as derived from mouth mount photo target data.

RHBOXP — Angular velocity of the head about the Y axis of the head anatomical coordinate system as derived from mouth mount photo target data.

RHCOXP — Angular velocity of the head about the Z axis of the head anatomical coordinate system as derived from mouth mount photo target data.

DAXSOP — The component of linear displacement of the head anatomical origin along the X axis of the sled coordinate system with respect to the sled coordinate system as derived from mouth mount photo target data. The sled coordinate system is aligned with the laboratory coordinate system and translates with the sled.

DAYSOP — The component of linear displacement of the head anatomical origin along the Y axis of the sled coordinate system with respect to the sled coordinate system as derived from mouth mount photo target data. The sled coordinate system is aligned with the laboratory coordinate system and translates with the sled.

DAZSOP — The component of linear displacement of the head anatomical origin along the Z axis of the sled coordinate system with respect to the sled coordinate system as derived from mouth mount photo target data. The sled coordinate system is aligned with the laboratory coordinate system and translates with the sled.

PHAOXP — Angle of rotation of the head about the X axis of the head anatomical coordinate system as derived from mouth mount photo target data. Head anatomical coordinate system is initially aligned with the laboratory coordinate system

PHBO2P — Same as PHAOXP except about the carried Y axis.

PHCO3P — Same as PHBO2P except about the carried Z axis.

EULER ANGLES

4HOO1P 4HOO2P 4HOO3P 4HOO4P	Quaternions - The four variables which define the angular orientation of the head anatomical coordinate system relative to the laboratory system as derived from mouth mount photo target data.
VNXSOP	The component of linear velocity of the T_1 anatomical origin along the X axis of the sled coordinate system with respect to the sled coordinate system as derived from T_1 mount photo target data. The sled coordinate system is aligned with the laboratory coordinate system and translates with the sled.
VNYSOP	The component of linear velocity of the T_1 anatomical origin along the Y axis of the sled coordinate system with respect to the sled coordinate system as derived from T_1 mount photo target data. The sled coordinate system is aligned with the laboratory coordinate system and translates with the sled.
VNZSOP	The component of linear velocity of the T_1 anatomical origin along the Z axis of the sled coordinate system with respect to the sled coordinate system as derived from T_1 mount photo target data. The sled coordinate system is aligned with the laboratory coordinate system and translates with the sled.
RNAOXP	Angular velocity of T_1 (first Thoracic vertebral body) about the X axis of the T_1 anatomical coordinate system as derived from T_1 mount photo target data.
RNBOXP	Angular velocity of T_1 (first Thoracic vertebral body) about the Y axis of the T_1 anatomical coordinate system as derived from T_1 mount photo target data.
RNCOXP	Angular velocity of T_1 (first Thoracic vertebral body) about the Z axis of the T_1 anatomical coordinate system as derived from T_1 mount photo target data.
DNXSOP	The component of linear displacement of the T_1 anatomical origin along the X axis of the sled coordinate system with respect to the sled coordinate system as derived from T_1 mount photo target data. The sled coordinate system is aligned with laboratory coordinate system and translates with the sled.
DNYSOP	The component of linear displacement of the T_1 anatomical origin along the Y axis of the sled coordinate system with respect to the sled coordinate system as derived from T_1 mount photo target data. The sled coordinate system is aligned with laboratory coordinate system and translates with the sled.
DNZSOP	The component of linear displacement of the T_1 anatomical origin along the Z axis of the sled coordinate system with respect to the sled coordinate system as derived from T_1 mount photo target data. The sled coordinate system is aligned with laboratory coordinate system and translates with the sled.

PNAOXP	Angle of rotation of T_1 (first Thoracic vertebral body) about the X axis of the T_1 anatomical coordinate system as derived from T_1 mount photo target data. The T_1 anatomical coordinate system is initially aligned with the laboratory coordinate system.	EULER ANGLES
PNBO2P	Same as PNAOXP except about the carried Y axis.	
PNCO3P	Same as PNBO2P except about the carried Z axis.	
4NOO1P 4NOO2P 4NOO3P 4NOO4P	Quaternions - The four variables which define the angular orientation of the T_1 anatomical coordinate system relative to the laboratory coordinate system as derived from T_1 mount photo target data.	
TIME	Time at which exposure occurred for film frames that were digitized to produce photo variables.	

The unit of all variables are consistent with

Linear measure - meters
Angular measure - radians
Time measure - seconds

APPENDIX B

DEFINITION OF CALCULATED VARIABLES BY THE HEAD AND NECK PROGRAMS

FORTRAN SYMBOL	DEFINITION
AGXP	The linear accleration of the head center of gravity along the x-axis of the head anatomical coordinate system with respect to the fixed laboratory coordinate system.
AGYP	The linear accleration of the head center of gravity along the y-axis of the head anatomical coordinate system with respect to the fixed laboratory coordinate system.
AGZP	The linear accleration of the head center of gravity along the z-axis of the head anatomical coordinate system with respect to the fixed laboratory coordinate system.
FOXLP	The force applied by the neck to the head parallel to the laboratory x-axis and passing through the occipital condylar point.
FOYLP	The force applied by the neck to the head parallel to the laboratory y-axis and passing through the occipital condylar point.
FOZLP	The force applied by the neck to the head parallel to the laboratory z-axis and passing through the occipital condylar point.
FOXP	The force applied by the neck to the head at the occipital condylar point parallel to the x-axis of the head anatomical coordinate system.
FOYP	The force applied by the neck to the head at the occipital condylar point parallel to the y-axis of the head anatomical coordinate system.
FOZP	The force applied by the neck to the head at the occipital condylar point parallel to the z-axis of the head anatomical coordinate system.
FOXTP	The force applied by the neck to the head parallel to the x-axis of the T_o coordinate system and passing through the occipital condylar point.
FOYTP	The force applied by the neck to the head along the y-axis of the T_o coordinate system and passing through the occipital condylar point.
FOZTP	The force applied by the neck to the head along the z-axis of the T_o coordinate system and passing through the occipital condylar point.
PSI	The angle between the y-axis of the T_o coordinate system and the projection of the y-axis of the head anatomical coordinate system onto the x-y plane of the T_o coordinate system.

FORTRAN SYMBOL	DEFINITION
RATIP	The distance from the head anatomical origin to the T anatomical origin as derived from photographic data.
ROTIP	The distance from the occipital condylar to the T anatomical origin.
TENTYP	The angle between the z-axis of the T_o coordinate system and the projection of the neck chord vector onto the x-z plane of the T_o coordinate system.
TENTXP	The angle between the neck chord vector and the x-z plane of the T_o coordinate system.
TETHNP	The angle between the y-axis of the head anatomical coordinate system and the projection of the y-axis of the T_o coordinate system onto the y-y plane of the head anatomical coordinate systems.
THTIYP	The angle between the z-axis of the T_o coordinate system and the projection of the z-axis of the head anatomical coordinate system onto the x-z plane of the T_o coordinate system.
THTIXP	The angle between the z-axis of the head anatomical coordinate system and the x-z plane of the T_o coordinate system.
TOXLP	The moment applied by the neck to the head about an axis parallel to the laboratory x-axis.
TOYLP	The moment applied by the neck to the head about an axis parallel to the laboratory y-axis.
TOZLP	The moment applied by the neck to the head about an axis parallel to the laboratory z-axis.
TOXP	The moment applied by the neck to the head axis that is parallel to the x-axis of the head anatomical coordinate system.
TOYP	The moment applied by the neck to the head axis that is parallel to the y-axis of the head anatomical coordinate system.
TOZP	The moment applied by the neck to the head axis that is parallel to the z-axis of the head anatomical coordinate system.
TOXTP	The moment applied by the neck to the head about an axis parallel to the z-axis of the T_o coordinate system.
TOYTP	The moment applied by the neck to the head about an axis parallel to the y-axis of the T_o coordinate system.
TOZTP	The moment applied by the neck to the head about an axis parallel to the z-axis of the T_o coordinate system.
T1XTP	The moment applied by the torso to the neck about an axis parallel to the x-axis of the T_o coordinate system.

FORTRAN SYMBOL	DEFINITION
T1YTP	The moment applied by the torso to the neck about an axis parallel to the y-axis of the T_o coordinate system.
T1ZTP	The moment applied by the torso to the neck about an axis parallel to the z-axis of the T_o coordinate system.

APPENDIX C

FORTRAN CODING OF THE DATA RETRIEVAL ANALYSIS AND DISPLAY SOFTWARE

The programs contained in this appendix are listed below. The general purpose programs listed are available on the NHTSA VAX in two locations; TSCPROGLIB and ASGPROGLIB.

COMP
XTRAC
XTRBLK
ACCDIS
DSPLAY
EXTRCT
DCIFER
DIRECT
STANDEV
STDCAL
SMEAN
MATH
MNMX
CURD
LABLE
SYMBOL
TITLE
DSP-WPAGE
DSP-OVERLAY
NCKNEW (NECK)
TRQPHOR (HEAD)

SUBROUTINE COMP

```
0001     *        SUBROUTINE COMP
0002     *        INCLUDE module for NCKPHO.FOR & TRQPHO.FOR
0003     *
0004              REAL DAXSOP(600),DAYSOP(600),DAZSOP(600),DNXSOP(600),
0005            &   DNYSOP(600),DNZSOP(600),PHAOXP(600),PHBO2P(600),PHCO3P(600),
0006            &   PNAOXP(600),PNBO2P(600),PNCO3P(600),TARRY(600),dcxsop(600)
0007     *
0008              COMMON /INDATA/ DAXSOP,DAYSOP,DAZSOP,DNXSOP,DNYSOP,DNZSOP,
0009            &   PHAOXP,PHBO2P,PHCO3P,PNAOXP,PNBO2P,PNCO3P,TARRY,DCXSOP
0010     *
```

SUBROUTINE XTRAC

```
      PROGRAM XTRAC
*  XTRAC:
*       This is the VAX version of the DEC-10 Neck Data Analysis
*       Package originally written for Curt Spenny.
*
*       VAX conversion and enhancement by Doug Gordon
*       Tweaking, repairs, and further enhancement by
*                                R. Stevens
*                                SDC

*       Parameters for DCIFER.FOR

      BYTE UNITS2(100)
      INTEGER*2 NUMB2(100)
      INTEGER INDX83(41),INDX93(44),INDX96(1),NUM,SUB,RECNUM,
     &    NMTMP(3),VARTMP,INDX44(4),INDX49(2),
     &    INDX60(2),INDX64(6),INDX65(5),INDX67(5),WRDTMP
      REAL A(801),B(801),C(801),MAX2(100),MIN2(100),SINIT
      CHARACTER*6 NAME2(100),RUN2(100)
      CHARACTER*5 LISTH(9)
      CHARACTER NEWNAM*40
      character dname*132
      character filnam*30,varnam*6

      INCLUDE 'XTRBLK/LIST'
      include 'dsppltblk.for/list'

      DATA LIST/'EXT','VMA','ADD','SUB','DIR','END','CLE',
     &          'FIL','CON','DIV','NOR','GET','DSP','STA'/
      DATA LIST2/'RUN','SUB'/
      DATA LIST3/'ALL'/
      DATA LIST4/'VAR'/

*       RUN NUMBERS

*       RUN NUMBERS

! Start of lateral test runs
      DATA LISTR/'LX1785','LX1793','LX1831','LX1860','LX1874',
     &    'LX1916','LX1960','LX1998','LX2010','LX2013','LX2027',
     &    'LX2032','LX2056','LX2060','LX2072','LX2090','LX2102',
     &    'LX2124','LX2137','LX2148','LX2151','LX2182','LX2282',
     &    'LX2294','LX2298','LX2302','LX2313','LX2326','LX2338',
     &    'LX2341','LX2355',
     $    'LX1454','LX1456','LX1457',
     &    'LX1458','LX1468','LX1470','LX1475','LX1484','LX1487',
     &    'LX1501','LX1503','LX1504','LX1505','LX1507','LX1509',
     &    'LX1510','LX1512','LX1513','LX1524','LX1525','LX1526',
     &    'LX1528','LX1471','LX1474',
! new nbdl lateral test runs
     &    'LX4050','LX4052','LX4053','LX4054','LX4055','LX4057',
     &    'LX4058','LX4059','LX4060','LX4068','LX4069','LX4070',
     &    'LX4071','LX4073','LX4074','LX4075','LX4076','LX4078',
     &    'LX4079','LX4080','LX4081','LX4083','LX4084','LX4085',
     &    'LX4088','LX4089','LX4090','LX4092','LX4093','LX4094',
```

```
     &    'LX4095','LX4097','LX4098','LX4099','LX4100','LX4104',
     &    'LX4107','LX4109','LX4110','LX4111','LX4115','LX4112',
     &    'LX4114','LX4116','LX4118','LX4119','LX4120','LX4123',
     &    'LX4124','LX4125','LX4126','LX4128','LX4129','LX4130',
     &    'LX4131','LX4133','LX4134','LX4135','LX4137','LX4138',
     &    'LX4139','LX4140','LX4142','LX4143','LX4144','LX4145',
     &    'LX4147','LX4148','LX4149','LX4151','LX4153','LX4155',
! Start of obliquetest runs
     &    'LX2763','LX2770','LX2772','LX2784',
     &    'LX2786','LX2799','LX2801','LX2813','LX2815','LX2827',
     &    'LX2829','LX2843','LX2872','LX2876','LX2916','LX2955',
     &    'LX2973','LX2979','LX2982','LX2985','LX2988','LX3049',
     &    'LX3053','LX3061','LX3065','LX3077','LX3085','LX3089',
     &    'LX3093','LX3097','LX3100','LX3102','LX3106','LX3122',
     &    'LX3129','LX3133','LX3145','LX3148','LX3153','LX3158',
     &    'LX3417',
! new nbdl oblique test runs
     &    'LX4159';'LX4161','LX4162','LX4163','LX4164','LX4166',
     &    'LX4167','LX4168','LX4170','LX4171','LX4172','LX4234',
     &    'LX4235','LX4236','LX4237','LX4238','LX4240','LX4241',
     &    'LX4242','LX4243','LX4244','LX4246','LX4247','LX4248',
     &    'LX4249','LX4251','LX4259','LX4260','LX4261','LX4263',
     &    'LX4264','LX4265','LX4266','LX4268','LX4269','LX4270',
     &    'LX4271','LX4276','LX4277','LX4280','LX4303','LX4281',
     &    'LX4282','LX4284','LX4286','LX4287','LX4288','LX4290',
     &    'LX4291','LX4292','LX4293','LX4295','LX4296','LX4297',
     &    'LX4298','LX4301','LX4302','LX4305','LX4306','LX4307',
     &    'LX4309','LX4310','LX4313','LX4314','LX4316',
! Start of frontal test runs
     &    'LX3524','LX3525','LX3530','LX3531','LX3536',
     &    'LX3537','LX3544','LX3548','LX3550','LX3558','LX3573',
     &    'LX3578','LX3583','LX3616','DOT307','DOT308','DOT309',
     &    'DOT310','DOT314','DOT331','DOT332','DOT333','DOT343',
     &    'DOT345',
! new nbdl frontal test runs
     &    'LX3801','LX3809','LX3824','LX3779','LX3780','LX3782',
     &    'LX3783','LX3785','LX3786','LX3788','LX3789','LX3791',
     &    'LX3793','LX3794','LX3796','LX3797','LX3798','LX3800',
     &    'LX3803','LX3804','LX3805','LX3807','LX3808','LX3812',
     &    'LX3814','LX3815','LX3817','LX3818','LX3819','LX3823',
     &    'LX3821','LX3822','LX3851','LX3852','LX3833','LX3837',
     &    'LX3839','LX3840','LX3841','LX3842','LX3872','LX3901',
     &    'LX3918','LX3854','LX3856','LX3880','LX3890','LX3857',
     &    'LX3858','LX3869','LX3870','LX3871','LX3875','LX3876',
     &    'LX3878','LX3882','LX3883','LX3885','LX3886','LX3887',
     &    'LX3889','LX3924','LX3893','LX3894','LX3895','LX3898',
     &    'LX3903','LX3900','LX3904','LX3906','LX3908','LX3909',
     &    'LX3913','LX3914','LX3916','LX3920','LX3921','LX3926',
     &    'LX3927','LX3928','LX3942','LX3953','LX3962','LX3958',
     &    'LX3939','LX3940','LX3941','LX3944','LX3945','LX3972',
     &    'LX3982','LX3946','LX3948','LX3949','LX3950','LX3951',
     &    'LX3954','LX3955','LX3957','LX3959','LX3961','LX3963',
     &    'LX3965','LX3968','LX3969','LX3970','LX3983','LX3985',
     &    'LX3986','LX3987','LX3989','LX3990','LX3991','LX3993',
     &    'LX3994','LX3995','LX3997','LX3998','LX3999',
```

```
!   new wsu data
      &     'DOT453','DOT454','T76008','DOT455'/

*         Subject Numbers
          DATA LISTH/'H0083','H0093','H0096','H0044',
      &           'H0049','H0060','H0064','H0065','H0067'/
*         Variable Names
          DATA LISTQ/ 'TIME','AAXXOS','AAYXOS','AAZXOS','QHAOXS',
      &    'QHBOXS','QHCOXS','VAXXOS','VAXSOS','VAYXOS','VAYSOS',
      &    'VAZXOS','VAZSOS','RHAOXS','RHBOXS','RHCOXS','DAXXOS',
      &    'DAXSOS','DAYXOS','DAYSOS','DAZXOS','DAZSOS','PHAOXS',
      &    'PHB02S','PHC03S','FH001S','FH002S','FH003S','FH004S',
      &    'ANXXOS','ANYXOS','ANZXOS','QNAOXS','QNBOXS','QNCOXS',
      &    'VNXXOS','VNXSOS','VNYXOS','VNYSOS','VNZXOS','VNZSOS',
      &    'RNAOXS','RNBOXS','RNCOXS','DNXXOS','DNXSOS','DNYXOS',
      &    'DNYSOS','DNZXOS','DNZSOS','PNAOXS','PNB02S','PNC03S',
      &    'FN001S','FN002S','FN003S','FN004S','ACXXOS','VCXXOS',
      &    'DCXXOS','VAXSOP','VAYSOP','VAZSOP','RHAOXP','RHBOXP',
      &    'RHCOXP','DAXSOP','DAYSOP','DAZSOP','PHAOXP','PHB02P',
      &    'PHC03P','FH001P','FH002P','FH003P','FH004P','VNXSOP',
      &    'VNYSOP','VNZSOP','RNAOXP','RNBOXP','RNCOXP','DNXSOP',
      &    'DNYSOP','DNZSOP','PNAOXP','PNB02P','PNC03P','DCXSOP',
      &    'FN002P','FN003P','FN004P'/
*         Subject Data
*   Subject 83
          DATA INDX83/1,2,3,4,7,8,10,11,14,16,17,18,19,20,26,30,32,
      &   34,36,38,40,42,44,48,49,50,51,54,56,57,58,60,62,63,67,68,
      &   70,73,75,77,79/
*   Subject 93
          DATA INDX93/5,6,9,12,13,15,21,22,23,24,27,28,29,31,33,35,
      &   37,39,41,43,45,46,47,52,53,55,59,61,64,65,66,69,71,72,74,
      &   76,78,80,81,82,83,84,85,86/
*   Subject 96
          DATA INDX96/25/
*   Subject 44
          DATA INDX44/90,98,103,108/
*   Subject 49
          DATA INDX49/92,110/
*   Subject 64
          DATA INDX64/88,109,96,100,104,105/
*   Subject 60
          DATA INDX60/91,93/
*   Subject 65
          DATA INDX65/87,95,99,102,107/
*   Subject 67
          DATA INDX67/89,94,97,101,106/

*         Array to indicate how to obtain data
*                 =0  Data is on file
*                 =1  Integrate once
*                 =2  Integrate twice
          DATA CREAT/7*0,1,0,1,0,1,4*0,2,1,2,1,2,1,13*0,1,0,1,0,1,4*0,
      &           2,1,2,1,2,1,8*0,1,2,3*1,13*0,3*1,13*0/
*         Gives variable number stored on file used to create values
          DATA ORIG/1,20,21,22,23,24,25,20,11,21,12,22,13,14,16,18,20,
```

```
     &    11,21,12,22,13,5,7,9,26,28,30,32,52,53,54,55,56,57,52,43,
     &    53,44,54,45,46,48,50,52,43,53,44,54,45,37,39,41,58,50,62,
     &    64,66,66,66,2,3,4,15,17,19,2,3,4,6,8,10,27,29,31,33,34,35,
     &    36,47,49,51,34,35,36,38,40,42,59,61,63,65/
*       Unit types for DSPLAY labels
        DATA UNITYP/4,3*3,3*7,6*2,3*8,6*1,3*5,4*9,3*3,3*7,6*2,3*8,
     &  6*1,3*5,4*9,3,2,1,3*2,3*8,3*1,3*5,4*9,3*2,3*8,3*1,3*5,4*9/
*       Flags for 0=Sensor 1=Photo
        DATA PHOTO/60*0,32*1/

***
***     START OF EXECUTABLE CODE
***
        KSTAT=LIB$INIT_TIMER()
        in_flags = 0
        call plot_questions(in_flags, dsplay$out_flags)
        WRITE(6,1000)
 1000   FORMAT(/23X,'Spenny Neck Data Analysis Package',/)
        CALL OUTPUT_DATE_TIME(78)
        PRINT *
        SUB=0
        ICNT=0
        IFPLAC=0
        ISUB=0
        RECNUM=0
        IBLANK=0
        OPEN(UNIT=1,FILE='SCRTCH',STATUS='UNKNOWN',ACCESS='DIRECT',
     &    MAXREC=MREC,RECL=RECLEN,ORGANIZATION='RELATIVE',ERR=05)
        READ(1,REC=1,ERR=10) NVAR,NAME,RUN,MAX,MIN,UNITS,NUMB
        CLOSE(UNIT=1)
   10   NEXTRD=1
        WRITE(6,1010)
 1010   FORMAT(1X,'XTR> ',$)
        IOPT1=0
        CALL DCIFER(3,14,LIST)
        IF(WHAT.EQ.1) THEN
          GOTO 10
        ELSE IF(WHAT.NE.5) THEN
          WRITE(6,1020)
 1020     FORMAT(1X,'Sorry - Cannnot identify this command',/1X,
     &    'Commands are:  EXTRACT   ADD       SUBTRACT  VMAGNITUDE',/1X
     &   ,'              CONSTANT  NORMALIZE DIVIDE    DSPLAY',/1X,
     &    '              DIRECTORY CLEAR     FILE      END',/1X,
     &    '              GET'/)
          WRITE(6,1030)
 1030     FORMAT(1X,'Please re-enter complete line'/)
          GOTO 10
        ENDIF
        OPEN(UNIT=1,FILE='SCRTCH',STATUS='UNKNOWN',ACCESS='DIRECT',
     &    MAXREC=MREC,RECL=RECLEN,ORGANIZATION='RELATIVE',ERR=05)
        NEXTRD=0
        GOTO(110,120,120,120,170,190,20,30,130,130,130,
     &  140,180,181),WRDNUM
        WRITE(6,1040)
 1040   FORMAT(/1X'[Internal Confusion...]'/)
```

```
      STOP  [Please Call a Programmer...]'

*     'CLEAR' Command

   20 NEXTRD=0
      CALL DCIFER(3,1,DUM)
      IF(WHAT.NE.2) NVAR=0
      IF(WHAT.EQ.2) NVAR=IVAL
      WRITE(1,REC=1) NVAR,NAME,RUN,MAX,MIN,UNITS,NUMB
      CLOSE(UNIT=1)
      GOTO 10

*     'FILE' Command:
   30 READ(1,REC=1) NVAR,NAME,RUN,MAX,MIN,UNITS,NUMB
      CALL DCIFER(3,1,DUMMY)
      IF(WHAT.NE.7) GOTO 100
      NEWNAM=IMAGE(FCHAR:LCHAR)
      OPEN(UNIT=20,FILE=NEWNAM,STATUS='UNKNOWN',ACCESS='DIRECT',
     &   MAXREC=MREC,RECL=RECLEN,ORGANIZATION='RELATIVE',ERR=100)
      WRITE(6,1050)
 1050 FORMAT(1X,'Variable #s or ALL> ',$)
      NEXTRD=1
      CALL DCIFER(3,1,LIST3)
!     print *,'what=',what
      IF(WHAT.EQ.5) THEN
        GOTO 80
      ELSE IF(WHAT.EQ.2) THEN
        NEXTRD=0
        GOTO 40
      ELSE
        WRITE(6,1060)
 1060   FORMAT(1X,'Variables must be input by directory number',/,
     &  1X,'  or ALL - FILE command ignored.'//)
        GOTO 70
      ENDIF
   40 READ(20,REC=1,ERR=50) NUMVAR,NAME2,RUN2,MAX2,MIN2,UNITS2,NUMB2
      GOTO 60
   50 NUMVAR=0
   60 IF(IVAL.GT.NVAR) THEN
        WRITE(6,1070) IVAL
 1070   FORMAT(1X,'Variable number ',I2,' exceeds number of ',
     &        'variables extracted - Check directory.',/1X,
     &        'Copy stopped at previous variable.'//)
        GOTO 70
      ENDIF
      KOUNT=NUMVAR+1
      if(kount.gt.99)then
         write(6,1071)kount
 1071    format(1x,'Variable number ',i3,'exceeds maximum number'/
     &           1x,'that can be filed (100).'/
     &           1x,'Copy stopped at previous variable.')
         goto 70
      endif
      RUN2(KOUNT)=RUN(IVAL)
      NAME2(KOUNT)=NAME(IVAL)
```

```
      MAX2(KOUNT)=MAX(IVAL)
      MIN2(KOUNT)=MIN(IVAL)
      UNITS2(KOUNT)=UNITS(IVAL)
      NUMB2(KOUNT)=NUMB(IVAL)
      READ(UNIT=1,REC=IVAL+1) (E(II),II=1,NUMB(IVAL))
      WRITE(20,REC=kount+1) (E(II),II=1,NUMB(IVAL))
      NUMVAR=NUMVAR+1
      ISAVQ=IVAL
      CALL DCIFER(3,1,DUMMY)
      IF(WHAT.EQ.1) THEN
        GOTO 70
      ELSE IF(WHAT.EQ.2) THEN
        GOTO 60
      ELSE
        WRITE(6,1080) ISAVQ
 1080   FORMAT(1X,'Illegal characters found in variable ',
     &          'specification line - Last variable transferred',/1X,
     &          'was ',I2//)
      ENDIF
   70 WRITE(20,REC=1) NUMVAR,NAME2,RUN2,MAX2,MIN2,UNITS2,NUMB2
      CLOSE(UNIT=20)
      CLOSE(UNIT=1)
      GOTO 10

*     ALL variables selected on FILE
*     Append to NEWNAM
   80 NUMVAR=0
      READ(20,REC=1,ERR=90) NUMVAR,NAME2,RUN2,MAX2,MIN2,UNITS2,NUMB2
   90 IF(NUMVAR+NVAR.GT.100) THEN
        WRITE(6,1090)
 1090   FORMAT(/1X,'Appending to the specified file will create ',
     &    'more than 100 variables.'/1x,'FILE command aborted - ',
     &    'Please choose another file...'/)
        WRITE(6,1030)
        GOTO 10
      ENDIF
      DO II=1,NVAR
        KOUNT=II+NUMVAR
        RUN2(KOUNT)=RUN(II)
        NAME2(KOUNT)=NAME(II)
        MAX2(KOUNT)=MAX(II)
        MIN2(KOUNT)=MIN(II)
        UNITS2(KOUNT)=UNITS(II)
        NUMB2(KOUNT)=NUMB(II)
        READ(UNIT=1,REC=II+1) (E(N),N=1,NUMB(II))
        WRITE(20,REC=kount+1) (E(N),N=1,NUMB(II))
      END DO
      N=NVAR+NUMVAR
      WRITE(20,REC=1) N,NAME2,RUN2,MAX2,MIN2,UNITS2,NUMB2
      CLOSE(UNIT=20)
      CLOSE(UNIT=1)
      GOTO 10

  100 WRITE(6,1100)
 1100 FORMAT(/1X,'An error has occurred in the file specification',
```

```
     &  /1X,'Please specify the file in quotes using VAX conventions'
        GOTO 10

*       EXTRACT command:
  110   CALL EXTRCT
        GOTO 10
*
*       First three MATH commands
*
  120   ICMD=WRDNUM-5                                  ! changed 1 to 5
        NEXTRD=0
        CALL MATH(ICMD)
        GOTO 10
*
*       Last three MATH commands
*
  130   ICMD=WRDNUM-5
        CALL MATH(ICMD)
        GOTO 10
*
*       The 'GET' command - read a SCRTCH format file into SCRTCH.DAT
*
  140   CALL DCIFER(9,1,DUMMY)
        IF(WHAT.NE.7) THEN
          WRITE(6,1100)
          WRITE(6,1030)
          GOTO 10
        ENDIF
        NEWNAM=IMAGE(FCHAR:LCHAR)
        OPEN(UNIT=21,FILE=NEWNAM,STATUS='OLD',ACCESS='DIRECT',
     &    MAXREC=MREC,RECL=RECLEN,ORGANIZATION='RELATIVE',ERR=150,
     &    readonly,defaultfile='xtrac$db:.dat')
        GOTO 160
  150   WRITE(6,1110) NEWNAM(1:LLEN(NEWNAM))
 1110   FORMAT(/1X,'The file ',A,' does not exist or is not in SCRTCH
     &  ' format.  GET command ignored.')
        GOTO 10
*
*       The file exists, so delete the current SCRTCH.DAT, and then
*       copy the new file in.
*
  160   CLOSE(UNIT=1,DISP='DELETE')
        OPEN(UNIT=1,FILE='SCRTCH',STATUS='NEW',ACCESS='DIRECT',
     &    MAXREC=MREC,RECL=RECLEN,ORGANIZATION='RELATIVE')
        READ(21,REC=1) NVAR,NAME,RUN,MAX,MIN,UNITS,NUMB
        WRITE(1,REC=1) NVAR,NAME,RUN,MAX,MIN,UNITS,NUMB
        DO JNK=2,NVAR+1
          READ(UNIT=21,REC=JNK)(E(II),II=1,NUMB(JNK-1))
          WRITE(1,REC=JNK)(E(II),II=1,NUMB(JNK-1))
        END DO
        CLOSE(UNIT=21)
        WRITE(6,1120) NEWNAM(1:LLEN(NEWNAM))
 1120   FORMAT(/1X,A,' successfully copied'/)
        GOTO 10
*
```

```
*       'DIR' command
*
  170   CALL DIRECT
        GOTO 10
*
*       DSPLAY subsystem - see documentation in DSPLAY.FOR
*
  180   CALL DSPLAY
        GOTO 10
*
*       'STA'
*
  181   filnam='scrtch.dat'
        image=image(ic:lchar)
        call baway(image)
        varnam=image(1:llen(image))
        call standev(filnam,varnam)
        goto 10
*
*
   05   call getdir(dname)
        write(6,1125)dname(1:llen(dname))
 1125   format(1x,'Error opening scratch file in directory :',a/
     &           1x,'Cannot continue execution.')
  190   WRITE(6,1130)
 1130   FORMAT(/1X,'End of execution - XTRAC'/)
        CALL LIB$SHOW_TIMER(,1)          ! Elapsed time
        CALL LIB$SHOW_TIMER(,2)          ! CPU time
        CALL LIB$SHOW_TIMER(,5)          ! Page faults
        END
```

SUBROUTINE XTRBLK

```
                SUBROUTINE XTRBLK
0001            BYTE UNITS(100),PHOTO(92)
0002            INTEGER*2 NUMB(100)
0003            INTEGER CREAT(92),ORIG(92),UNITYP(92),RECLEN,MREC,NVAR,WHAT,
0004          &   FCHAR,WRDNUM,IVAL
0005            REAL MAX(100),MIN(100),E(598),TARRAY(598)
0006            CHARACTER*6 NAME(100),RUN(100),LISTR(380),LISTQ(92)
0007            CHARACTER*3 LIST(14),LIST2(2),LIST3,LIST4
0008            CHARACTER*10 XRUN,ID,TEST,IMAGE*80
0009
0010            COMMON /INPUT/  IMAGE,NEXTRD,IC,FCHAR,LCHAR,WHAT,LSTPOS,
0011          &                 WRDNUM,IVAL,VALUE
0012            COMMON /HEADER/ NVAR,NAME,RUN,MAX,MIN,UNITS,NUMB
0013            COMMON /INPEXT/ TARRAY,E
0014            COMMON /PHOTON/ PHOTO
0015            COMMON /DICTIO/ LISTR,LISTQ,LIST,LIST2,LIST3,LIST4
0016            COMMON /VARATT/ CREAT,ORIG,UNITYP
0017
0018            PARAMETER (MREC=101)
0019            PARAMETER (RECLEN=598)
0020
```

SUBROUTINE ACCDIS

```
0001
0002    ***********************************************************************
0003    ***********************************************************************
0004          SUBROUTINE ACCDIS(SINIT,B,A,VINIT,IOPT1,PHOTO,NUM,TARRAY)
0005    *  ACCDIS:                                    Douglas A. Gordon
0006    *                                             Arcon Corporation
0007    *
0008    *       Calculate velocities and displacements from
0009    *       accelerations (sensor data) or velocities from
0010    *       displacements (photographic data)
0011    *
0012    *     - A(i) is input (accelerations or displacements)
0013    *       V(i) is requested output in REAL*8 format
0014    *       B(i) is requested outut returned as REAL*4
0015    *       TARRAY(i) is variable time step array for photo data
0016    *       SINIT is initial displacement @ time T=0
0017    *       VINIT is initial velocity @ time T=0
0018    *       IOPT1 = 1, calculate velocities
0019    *             = 2, calculate displacements
0020    *       PHOTO = 1, for photographic data (variable time step)
0021    *
0022          BYTE PHOTO
0023          REAL A(NUM),B(NUM),TARRAY(NUM)
0024          REAL*8 DELT,V(801),S(801),q,q1
0025
0026          K=NUM-1
0027          IF(PHOTO.EQ.1) GOTO 3
0028          DELT=0.0005/2.
0029          IF(IOPT1.EQ.2) GOTO 2
0030    *
0031    *       Calculate velocities from accelerations.  (sensor)
0032    *
0033          V(1)=DBLE(VINIT)
0034
0035          DO I=1,K
0036            V(I+1)=V(I)+DELT*DBLE((A(I)+A(I+1)))
0037          END DO
0038          GOTO 5
0039    *
0040    *       Calculate displacements from accelerations (sensor)
0041    *
0042        2   V(1)=DBLE(SINIT)
0043          S(1)=DBLE(VINIT)
0044          DO I=1,K
0045            II=I+1
0046            Q=DELT*DBLE((A(I)+A(II)))
0047            S(II)=S(I)+Q
0048            Q1=DELT*(S(I)+S(II))
0049            V(II)=V(I)+Q1
0050          END DO
0051          GOTO 5
0052    *
0053    *       Calculate velocities from displacements with variable time
0054    *       step (photo)
0055    *
0056        3   V(1)=VINIT
0057          DO I=1,K
```

```
0058                DELT=TARRAY(I+1)-TARRAY(I)
0059                V(I+1)=(A(I+1)-A(I))/DELT
0060              END DO
0061      *
0062      *       Copy REAL*8 array V into REAL*4 array B for return to main
0063      *       program
0064      *
0065           5  DO I=1,NUM
0066                B(I)=V(I)
0067              END DO
0068              RETURN
0069              END
```

SUBROUTINE DSPLAY

```
************************************************************************
************************************************************************
      SUBROUTINE DSPLAY
*  DSP:                                        Douglas A. Gordon
*                                              Arcon Corporation
*
*     Plotting  module of  the Spenny  Neck Analysis  Package.
*     Currently supported with the TEKTRONIX  TCS package, the
*     original used  DISSPLA, and  will never  be converted to
*     that in the future.  There is no 3-D plotting under TCS.
*
      integer*4 dsplay$num_cmds
      parameter (dsplay$num_cmds = 11)

      byte already_lasered/.false./, plotted_something, delete_it,
     &   copy_it
      INTEGER*4 READ1,READ2,XTMPLB,YTMPLB,ZTMPLB,I3READ(4),RECLN,
     &   RECNM,PLST(20,4),PLST3(20,5),N3PNT(3),NPLOT,N3PLOT,KPLOT,
     &   BAUDQ, points(20), out_flags, tt_lun, submit_laser_file
      REAL XMAX2,XMAX3,XMIN2,XMIN3,YMIN2,YMIN3,YMAX2,YMAX3,ZMAX3,
     &   ZMIN3,ARRAY1(598),ARRAY2(598),ARRAY3(598),xplt(598,20),
     &   yplt(598,20)
      CHARACTER TTL*30,dans*1, laser_file*252
      CHARACTER*3 LISTD(dsplay$num_cmds),LISTD1,LISTD2
      CHARACTER*10 RNTMP,XLABEL,YLABEL,ZLABEL,DUMMY,NAMTMP,
     &   LABEL(0:9),rntmps(20)
      byte quit

      INCLUDE 'XTRBLK/LIST'
      include 'pltdef.for/list'
      include 'dsppltblk.for/list'
*
      DATA LISTD/'DIR','PLO','PL3','DIS','END','XTR','COP','XSC',
     &   'YSC', 'LAS', 'FIG'/
      DATA DUMMY/'          '/
      DATA LISTD1/'RUN'/
      DATA LISTD2/'TIM'/
      DATA LABEL/' ','METERS','M/SEC','M/SEC*SEC','TIME','RADIANS',
     &           'NEWTON-M','RAD/SEC**2','RADIAN/SEC','NEWTONS'/
***   START OF EXECUTABLE CODE
      out_flags = plt$m_box + plt$m_xline + plt$m_yline + plt$m_nosym
      delete_it = .true.
      copy_it = .false.
*       The scratch file is opened and the dictionary information
*       is read from the first record.
*       NVAR = number of variables
*       NAME,
*       RUN  = arrays of 100 6 character fields, var names & runs
*       MAX,
*       MIN  = arrays of 100 reals, min or max for each NAME
*       UNITS = array of 100 bytes containing #'s 1-7
*       NUMB  = array of number of points for the NAME list
      OPEN(UNIT=1,FILE='SCRTCH',STATUS='OLD',ACCESS='DIRECT',
     &   MAXREC=MREC,RECL=RECLEN,ORGANIZATION='RELATIVE',
     &   IOSTAT=IOS)
```

```
      READ(1,REC=1,IOSTAT=IOS) NVAR,NAME,RUN,MAX,MIN,UNITS,
     &    NUMB
      MARK=0
      NPLOT=0
      IVARN=0
      IWORD=0
      RECNM=0
      N3PLOT=0
      KPLOT=0
      quit=.false.
*
   10 NEXTRD=1                                  ! read new card, next field!
      WRITE(6,1000)
 1000 FORMAT(1X,'DSP> ',$)
      CALL DCIFER(3,dsplay$num_cmds,LISTD)
      NEXTRD=0                                  ! read same card, next field!
      IF(WHAT.EQ.5)
     &  GOTO(20,30,240,250,690,680,251,3000,3010,3050,3030),WRDNUM

      IF(WHAT.EQ.1) GOTO 10                     ! if end of card!
      WRITE(6,1010)                             ! otherwise...!
      WRITE(6,1020)
 1010 FORMAT(1X,'Sorry - cannot identify this command',/
     &  1x,'Commands are:  PLOT    PL3D    DIRECTORY  DISPLAY',/
     &  1x,'               XTRAC   COPY    XSCALE     YSCALE',/
     &  1x,'               LASER   FIGURE  END'/)
 1020 FORMAT(1X,'Please re-enter complete line',/,1x)
      GOTO 10
*****************************************************************
***  'DIRECTORY'
*
   20 CALL DIRECT
      GOTO 10
*****************************************************************
***  'PLOT'
*
   30 CALL DCIFER(3,1,LISTD1)                   ! look for RUN following PLOT!
      IF(WHAT.NE.5) then
         rntmp=dummy
         ic=lstpos
      else
         CALL DCIFER(6,NVAR,RUN)                ! check for valid run in dic.!
         if(what.eq.5)then
            rntmp=run(wrdnum)
         else
            WRITE(6,1030)
            WRITE(6,1020)
 1030       FORMAT(/1X,'Run number must follow the identifier RUN'/)
            goto 10
         endif
      endif
*
   60 call dcifer(3,1,listd2)
      if(what.eq.5)then                         ! first var is time
        read1=999
```

```
  call dcifer(3,1,listd2)
  if(what.eq.5)then                        ! second var is time
    read2=999
    goto 110
  else                                     ! second var not time
    ic=lstpos
    do j=1,nvar
      if(run(j).eq.rntmp)then              ! if same run #
        namtmp=name(j)
        call dcifer(6,1,namtmp)            ! check for right var
        if(what.eq.5)goto 100              ! right var
        ic=lstpos                          ! keep looking
      endif
    enddo
    write(6,1040)                          ! 2nd not in scrtch
    write(6,1050)
    write(6,1020)
    goto 10
  endif
else                                       ! first var not time
  ic=lstpos
  do i=1,nvar
    if(run(i).eq.rntmp)then                ! check for right run
      namtmp=name(i)
      call dcifer(6,1,namtmp)              ! check for right var
      if(what.eq.5)then                    ! if right var
        read1=i
        call dcifer(3,1,listd2)            ! second var time ?
        if(what.eq.5)then                  ! it is time
          read2=999
          goto 110
        else                               ! it isn't time
          ic=lstpos
          do j=1,nvar                      ! look for var in scrtch
            if(run(j).eq.rntmp)then
              namtmp=name(j)
              call dcifer(6,1,namtmp)
              if(what.eq.5) goto 100
              ic=lstpos
            endif
          enddo
          write(6,1040)
          write(6,1050)
          write(6,1020)
          goto 10
        endif
        ic=lstpos
      endif
      ic=lstpos
    endif
  enddo
  write(6,1040)
  write(6,1050)
  write(6,1020)
  goto 10
```

```
        endif
 1040   format(/1x,'Input processing shows a run/variable mismatch'/)
 1050   format(1x,'For run/variable info use the DIRECTORY command'/)
*
  100   READ2=J
*
  110   IF(READ1.NE.READ2) GOTO 120
        WRITE(6,1060)
        WRITE(6,1020)
 1060   FORMAT(/1x,'There really is no sense in plotting the same
     &  variables')
        GOTO 10
*
  120   CALL DCIFER(3,1,DUMMY)
        if(what.ne.2.and.what.ne.1)then         ! non-integer marker
           write(6,1069)
 1069   format(1x,'An integer is required for marker value (0-8)'
     &/1x,'Please try again...',/)
           goto 10
        else IF(WHAT.EQ.2.AND.IVAL.LE.8)then    ! a proper marker value
           goto 130
        else if(what.eq.2.and.ival.gt.8)then    ! marker values only to 8
                WRITE(6,1070)
 1070   FORMAT(1X,'Marker values are in the range of 0-8 (integer)'
     & /1x,'Please try again...',/)
                goto 10
        else                                    ! value blank, use 0
                IVAL=0
        endif
*
  130   MARKER=IVAL
        IF(NPLOT.LE.0) GOTO 140
*
*       This section is commented out to allow plotting of
*       different variables.
*
  140   NPNT1=0
        NPNT2=0
  150   IF(READ1.NE.999) NPNT1=NUMB(READ1)
        IF(READ2.NE.999) NPNT2=NUMB(READ2)
        IF(READ1.NE.999.AND.READ2.NE.999) GOTO 190
*
*  ARRIVED HERE BECAUSE ONE VARIABLE IS 'TIME'
*
        IF(NPNT1.NE.598.AND.NPNT2.NE.598) GOTO 160
        NPNT1=598
        NPNT2=598
        GOTO 190
*
  160   DO 170 IJ=1,NVAR
        IF((NAME(IJ).EQ.'TIME'.AND.RUN(IJ).EQ.RNTMP).or.
     &    (rntmp.eq.dummy)) GOTO 180
  170   CONTINUE
        WRITE(6,1100)
        GOTO 10
```

```
*
  180   IF(READ1.EQ.999) read1=IJ
        IF(READ2.EQ.999) READ2=IJ
        GOTO 150
*
*       CHECK  FOR  BOUNDED PLOT
*
  190   CALL DCIFER(3,1,DUM)
        IF(WHAT.EQ.1) GOTO 210
        IF(WHAT.NE.2) GOTO 200
        NPNT1=IVAL
        READ(1,REC=READ1+1)ARRAY1
        CALL MNMX(ARRAY1,MIN,MAX,READ1,NPNT1)
        NPNT2=IVAL -
        READ(1,REC=READ2+1)ARRAY1
        CALL MNMX(ARRAY1,MIN,MAX,READ2,NPNT2)
        WRITE(1,REC=1)NVAR,NAME,RUN,MAX,MIN,UNITS,NUMB
        GOTO 210
*
  200   WRITE(6,1080)
 1080   FORMAT(1X,'Unrecognized user input following plot symbol',
     &' ignored')
*
  210   IF(NPNT1.EQ.NPNT2) GOTO 220
        WRITE(6,1090)
 1090   FORMAT(1X,'A mismatch between the number of points '
     &,/,' to be plotted has been discovered.'/,
     & ' the PLOT command will be ignored !',/)
        GOTO 10
*
  220   NPNT=NPNT1
        NPLOT=NPLOT+1
        rntmps(nplot)=rntmp
        PLST(NPLOT,1)=READ1
        PLST(NPLOT,2)=READ2
        PLST(NPLOT,3)=MARKER
        PLST(NPLOT,4)=NPNT
  230   GOTO 10
 1100   format(1x,'TIME variable for photographic data is not in the',/
     &' SCRATCH file directory',//'PLOT request is ignored',/)
**********************************************************************
***   'DISPLAY'
*
  250   IF(NPLOT.EQ.0) GOTO 640
        IF(PLST(1,1).EQ.999) THEN
          dsplay$xmin = 0.0
          dsplay$xmax = 0.3
          dsplay$incx = 3
          out_flags = out_flags + plt$m_xscale
        ENDIF
        IF(PLST(1,2).EQ.999) THEN
          dsplay$ymin = 0.0
          dsplay$ymax = 0.3
          dsplay$incy = 3
          out_flags = out_flags + plt$m_yscale
```

```
      ENDIF
*
*     Start the plot
*
      TTL(21:30)=' '
      ITLEN=20
      IF(PLST(1,1).EQ.999) THEN
        TTL(1:10)='TIME      '
      ELSE
        TTL(1:10)=NAME(PLST(1,1))
      ENDIF
      IF(PLST(1,2).EQ.999) THEN
        TTL(11:20)='TIME      '
      ELSE
        TTL(11:20)=NAME(PLST(1,2))
      ENDIF
      IF(NPLOT.EQ.1) THEN
        ITLEN=30
        IF(PLST(1,1).NE.999) THEN
          TTL(21:30)=RUN(PLST(1,1))
        ELSEIF(PLST(1,2).NE.999) THEN
          TTL(21:30)=RUN(PLST(1,2))
        ELSE
          ITLEN=20
        ENDIF
      ENDIF
  260 IF(PLST(1,1).EQ.999) THEN
        XTMPLB=4
      ELSE
        XTMPLB=UNITS(PLST(1,1))
      ENDIF

      IF(PLST(1,2).EQ.999) THEN
        YTMPLB=4
      ELSE
        YTMPLB=UNITS(PLST(1,2))
      ENDIF
      XLABEL=LABEL(XTMPLB)
      YLABEL=LABEL(YTMPLB)

      YPOS=9.2

      quit=.false.
      DO IQ=1,NPLOT
        IF(PLST(IQ,1).EQ.999) GOTO 270
        RECNM=PLST(IQ,1)+1
        READ(1,REC=RECNM)ARRAY1
        GOTO 290
  270   ARRAY1(1)=0.0

        TIME=.0005
        DO IK=2,598
          ARRAY1(IK)=ARRAY1(IK-1)+TIME
        end do
```

```
290       IF(PLST(IQ,2).EQ.999) GOTO 300
          RECNM=PLST(IQ,2)+1
          READ(1,REC=RECNM)ARRAY2
          GOTO 320

300       ARRAY2(1)=0.0
          TIME=.0005
          DO IS=2,598
            ARRAY2(IS)=ARRAY2(IS-1)+TIME
          end do
320       call lib$movc3(2392, array1, xplt(1,iq))
          call lib$movc3(2392, array2, yplt(1,iq))
          points(iq) = plst(iq,4)
        end do

        if(already_lasered) then
          ttl = ' '
          xlabel = ' '
          ylabel = ' '
        else
          if((dsplay$out_flags .and. plt$m_vt240) .ne. 0)
     &      call vt200_set_mode(4)
        endif

        call dsp_overlay(xplt, yplt, points, nplot, 598, 0, ttl,
     &    xlabel, ylabel, out_flags)

        do iq = 1, nplot
          if(plst(iq,1).eq.999)then
            call lable(rntmps(iq),'TIME  ',name(plst(iq,2)),
     &        quit,plst(iq,3))
          else if(plst(iq,2).eq.999)then
            call lable(rntmps(iq),name(plst(iq,1)),'TIME  ',
     &        quit,plst(iq,3))
          else
            call lable(rntmps(iq),name(plst(iq,1)),
     &        name(plst(iq,2)),quit,plst(iq,3))
          endif

          close(unit=87)
        end do
        quit=.true.
        if(read1 .eq. 999) read1 = 1
        if(read2 .eq. 999) read2 = 1
        call lable(rntmp,name(read1),name(read2),quit)
        if(.not. already_lasered) then
          if(copy_it) then
            call hdcopy
          else
            call plhold
          endif
          call newpag
          if((dsplay$out_flags .and. plt$m_vt240) .ne. 0) then
            call vt200_set_mode(5)
          endif
```

```
        endif
        plotted_something = .true.
        out_flags = plt$m_box + plt$m_xline + plt$m_yline + plt$m_nosym
        copy_it = .false.
        GOTO 660
*****************************************************************************
***  'XSCALE'
*
 3000   its_x = .true.
        goto 3020
*****************************************************************************
***  'YSCALE'
*
 3010   its_x = .false.
 3020   call dcifer(3,1,dummy)                    ! min axis value
        if(what .eq. 3) then
          qtmp1 = value
        else
          goto 4000
        endif

        call dcifer(3,1,dummy)                    ! max axis value
        if(what .eq. 3) then
          qtmp2 = value
        else
          goto 4000
        endif

        call dcifer(3,1,dummy)                    ! num tic marks
        if(what .eq. 2) then
          iqtmp = ival
        else
          goto 4000
        endif

        if(its_x) then
          dsplay$xmin = qtmp1
          dsplay$xmax = qtmp2
          dsplay$incx = iqtmp
          out_flags = out_flags + plt$m_xscale
        else
          dsplay$ymin = qtmp1
          dsplay$ymax = qtmp2
          dsplay$incy = iqtmp
          out_flags = out_flags + plt$m_yscale
        endif
        goto 10
 4000   write(6,4010)
 4010   format(1x,'Format for XSCALE & YSCALE is:'/
     &    1x,'COMMAND <min-value> <max-value> <tic-marks>')
        write(6,1020)
        goto 10
*****************************************************************************
***  'FIGURE'
*
```

```
 3030    call dcifer(32,1,dummy)
         if(what .eq. 7) then
           dsplay$figure = image(fchar:lchar)
         else
           write(6,3040)
 3040      format(1x,'Figure title must be enclosed in single quotes')
           write(6,1020)
         endif
         goto 10
*****************************************************************************
***  'LASER'
*
 3050    if(already_lasered) then
           write(6,3060)
 3060      format(1x,'The laser command has already been issued')
           goto 10
         endif
         already_lasered = .true.
         out_flags = out_flags + plt$m_laser
         call setup_laser_file(' ', laser_file)
         write(6,3070) laser_file(1:llen(laser_file))
 3070    format(1x,'Output plot file is ',a)
         plotted_something = .false.
         goto 10
*****************************************************************************
***  'copy'
*
  251    copy_it = .true.
         goto 250
*
  640    write(6,1120)
 1120    format(/1x,'Nothing to plot ! ')
         nplot=0
         goto 10
*
  660    NPLOT=0
         N3PLOT=0
         KPLOT=KPLOT+1
         IF(KPLOT.GE.2) KPLOT=0
         GOTO 10
*****************************************************************************
***  'PL3D'
*
  240    print *
         print *, '3-D plotting not currently supported'
         print *
         goto 10

  670    WRITE(6,1130)
 1130    FORMAT(/1X,'Error OPENing or READing SCRTCH -')
         CALL IOSMSG(IOS)
  680    WRITE(6,1140)
 1140    FORMAT(/1X,'Returning to XTRAC'/)
         RETURN
  690    if(already_lasered) then
```

```
          if(plotted_something) then
            istat = submit_laser_file('SYS$MANAGER:TEKTRONIX.LIS',
     &        laser_file, delete_it)
          else
            inquire(file='tt',number=tt_lun)
            close(unit=tt_lun,disp='delete')
          endif
        endif
        WRITE(6,1150)
1150    FORMAT(/1X,'End of execution - XTRAC/DSPLAY'/)
        CALL LIB$SHOW_TIMER()
        call sys$exit(%val(1))
        END
```

SUBROUTINE EXTRCT

```
**********************************************************************
**********************************************************************
      SUBROUTINE EXTRCT
*  EXTRCT:                                          Douglas A. Gordon
*                                                   Arcon Corporation
*
*       The  EXTRACT command  for XTRAC.FOR.   This  is the  data  read
*       routine for the Spenny  Neck Analysis package.  The  .EXT files
*       are binary files  containing 66 of  the original 92  variables.
*       Missing   variables   are  obtained   through   integration  or
*       differentiation of existing variables.
*
      REAL A(598)
      CHARACTER FILENM*10

      INCLUDE 'XTRBLK/LIST'

*
*       Check next word
*
      NEXTRD=0
      CALL DCIFER(3,2,LIST2)
      IF(WHAT.NE.5) THEN
        WRITE(6,1000)
 1000     FORMAT(1X,'The word RUN or SUB must follow EXTRACT'/)
        WRITE(6,1010)
 1010     FORMAT(1X,'Please re-enter complete line')
        RETURN
      ENDIF
*  word was run or sub - check to see if next word is ALL
      NWORD=WRDNUM
      CALL DCIFER(3,1,LIST3)
      IF(WHAT.EQ.5) THEN
        IRUN=999
        ISUB=0
        GOTO 30
      ENDIF
      IC=LSTPOS
      IF(NWORD.EQ.2) GOTO 10
      CALL DCIFER(6,380,LISTR)        !Check for c rrect run #
      IF(WHAT.NE.5) THEN
        WRITE(6,1020)
 1020     FORMAT(1X,'The run number is not valid')
        WRITE(6,1010)
        RETURN
      ENDIF
      IFPLAC=WRDNUM
      GOTO 20

*  check subject numbers
   10 CALL DCIFER(6,9,LISTH)
      IF(WHAT.NE.5) THEN
        WRITE(6,1030)
 1030     FORMAT(1X,'The subject number is invalid')
        WRITE(6,1010)
```

```
          RETURN
        ENDIF
   20   ISUB=WRDNUM
        IF(IRUN.EQ.999) ISUB=0
   30   CALL DCIFER(3,1,LIST4)
        IF(WHAT.NE.5) THEN
          WRITE(6,1040)
 1040     FORMAT(1X,'Keyword VAR must follow RUN or SUB')
          WRITE(6,1010)
          RETURN
        ENDIF
*  ALL or variable name
        CALL DCIFER(3,1,LIST3)
        IF(WHAT.EQ.5) THEN
          RECNUM=999
          GOTO 50
        ENDIF
   40   IC=LSTPOS
        CALL DCIFER(6,92,LISTQ)
        IF(WHAT.NE.5) THEN
          WRITE(6,1050)
 1050   FORMAT(1X,'The variable name is not in the list of legal',
     &  ' variables.'/)
          WRITE(6,1010)
          RETURN
        ENDIF
        IVNM=WRDNUM
        IF(CREAT(WRDNUM).NE.0) IOPT1=CREAT(WRDNUM)
        RECNUM=ORIG(WRDNUM)*2-1
   50   IF(NWORD.EQ.2) GOTO 90           !Select by subject
   60   IF(IRUN.EQ.999) IFPLAC=IFPLAC+1
        FILENM=LISTR(IFPLAC)//'.EXT'

        OPEN(UNIT=20,FILE=FILENM,ACCESS='SEQUENTIAL',STATUS='OLD',
     &    FORM='UNFORMATTED',ERR=80,IOSTAT=JXS,READONLY,
     &    DEFAULTFILE='xtrac$dat:')

        READ(20) NUM,XRUN,ID,TEST,SINIT,VINIT
        NUM=MIN0(NUM,598)        !<<< Mod for Curt Spenny
        READ(20) (TARRAY(I),I=1,NUM)
        REWIND 20
        IF(RECNUM.EQ.999) GOTO 70
        IF(RECNUM.NE.1) THEN
          DO IB=1,RECNUM-1
            READ(20)
          END DO
        ENDIF
        READ(20) NUM,XRUN,ID,TEST,SINIT,VINIT
*         print 3000,nvar,num,xrun,id,test,sinit,vinit
        NUM=MIN0(NUM,598)        !<<<<<<<<<<<<<<<<<<<
        READ(20) (E(II),II=1,NUM)
        CLOSE(UNIT=20)
        IOPT1=CREAT(IVNM)
*       print *,'IOPT1 =',iopt1,'  IVNM =',ivnm
        IF(IOPT1.NE.0) THEN
```

```
            CALL ACCDIS(SINIT,A,E,VINIT,IOPT1,PHOTO(IVNM),NUM,TARRAY)
            DO ITX=1,NUM
              E(ITX)=A(ITX)
            END DO
          ENDIF
          OPEN(UNIT=1,FILE='SCRTCH',STATUS='UNKNOWN',ACCESS='DIRECT',
         &   RECL=RECLEN,ORGANIZATION='RELATIVE',ERR=80,IOSTAT=JXS,
         &   MAXREC=MREC)
          NVAR=NVAR+1
          if(nvar.gt.100)then
            write(6,1065)name(nvar-1)
 1065       format(1x,'Attempt to EXTRACT more than 100 variables -',
         &     /1x,'EXTRACT stopped at variable ',a/)
            nvar=nvar-1
            return
          endif
          NAME(NVAR)=LISTQ(IVNM)
          RUN(NVAR)=LISTR(IFPLAC)
          CALL MNMX(E,MIN,MAX,NVAR,NUM)
          UNITS(NVAR)=UNITYP(IVNM)
          NUMB(NVAR)=NUM
          WRITE(1,REC=1) NVAR,NAME,RUN,MAX,MIN,UNITS,NUMB
          IREC=NVAR+1
          WRITE(1,REC=IREC) (E(II),II=1,NUMB(NVAR))
          IF(IRUN.EQ.999.AND.IFPLAC.NE.92) GOTO 60
          RETURN
    70    OPEN(UNIT=1,FILE='SCRTCH',STATUS='UNKNOWN',ACCESS='DIRECT',
         &   RECL=RECLEN,ORGANIZATION='RELATIVE',ERR=80,IOSTAT=JXS,
         &   MAXREC=MREC)
          PRINT *
          DO IDD=1,92
            RECNUM=ORIG(IDD)*2-1
            IF(ORIG(IDD).NE.1) THEN
              DO IEX=1,RECNUM-1
                READ(20)
              END DO
            ENDIF
            READ(20) NUM,XRUN,ID,TEST,SINIT,VINIT
!           print 3000,idd,num,xrun,id,test,sinit,vinit
 3000       format(1x,i2,')',i6,1x,3a10,2g)
            NUM=MIN0(NUM,598)
            READ(20) (E(II),II=1,NUM)
            WRITE(6,1060) '*'
 1060       FORMAT(1H+,A1,$)
            REWIND 20
            IF(CREAT(IDD).NE.0) THEN
              IOPT1=CREAT(IDD)
!           print *,'Calculating variable ',listq(idd)
              CALL ACCDIS(SINIT,A,E,VINIT,IOPT1,PHOTO(IDD),NUM,TARRAY)
              DO IXZ=1,NUM
                E(IXZ)=A(IXZ)
              END DO
            ENDIF
            NVAR=NVAR+1
            IF(NVAR.GT.100) THEN
```

```
          WRITE(6,1070) name(nvar-1)
1070      FORMAT(/1X,'Attempt to EXTRACT more than 100 variables -',
     &       /1x,'EXTRACT stopped at variable ',a/)
          RETURN
        ENDIF
        NAME(NVAR)=LISTQ(IDD)
        RUN(NVAR)=LISTR(IFPLAC)
        CALL MNMX(E,MIN,MAX,NVAR,NUM)
        UNITS(NVAR)=UNITYP(IDD)
        NUMB(NVAR)=NUM
        WRITE(1,REC=1)NVAR,NAME,RUN,MAX,MIN,UNITS,NUMB
      IREC=NVAR+1
      WRITE(1,REC=IREC) (E(I),I=1,NUMB(NVAR))
      END DO
      IF(IFPLAC.NE.110.AND.IRUN.EQ.999) GOTO 60
      RETURN
 80   print *,'OPEN ERROR'
      call iosmsg(jxs)
      return
 90   PRINT *, 'No select by subject now...'
      RETURN
      END
```

SUBROUTINE DCIFER

```
0001            SUBROUTINE DCIFER(NCLST,NDICT,LIST)
0002      *  DCIFER:                                          Douglas A. Gordon
0003      *                                                   Arcon Corporation
0004      *
0005      *       Adapted for the VAX from the similar package on the TSC
0006      *       DEC-10.  Original author(s) unknown.
0007      *
0008      *       This is the command parser for the NECK DATA ANALYSIS
0009      *       PACKAGE originally written for C. Spenny.  See XTRAC.
0010      *
0011            - FUNCTIONS & SUBROUTINES CALLED:
0012
0013                            CMPRES          External Library
0014                            CTB             External Library
0015                  L*1       ISDIG           External Library
0016      *           I*4       LLEN            External Library
0017      *                     STRINP          External Library
0018      *           I*4       TOINT           External Library
0019      *           R*4       TOREAL2         External Library
0020      *                     TTB             External Library
0021      *
0022            BYTE IFDP,ISDIG
0023            INTEGER WHAT,FCHAR,WRDNUM,IVAL,LLEN,TOINT,GET_STRING
0024            CHARACTER*(*) LIST(NDICT)
0025            CHARACTER WORD*10,IMAGE*80,SQUOTE*1,DELIM*3
0026      *
0027      *       In common block INPUT:
0028      *               IMAGE   =  80 character command "card"
0029      *               NEXTRD  =  0, Read next field on this card
0030      *                          1, Read new card, first field
0031      *                          2, Reread first field on same card
0032      *               IC      =  Pointer to current card column
0033      *               FCHAR   =  Location of first char in string
0034      *               LCHAR   =  Location of last char in string
0035      *               WHAT    =  1, End of card
0036      *                          2, Integer (returned in IVAL)
0037      *                          3, Real number (returned in VALUE)
0038      *                          4, Word, not in dictionary
0039      *                          5, Word, in dictionary
0040      *                          6, End-of-file condition
0041      *                          7, Character string
0042      *                          8, Illegal char in numeric field
0043      *               LSTPOS  =  Beginning location of last field read
0044      *               WRDNUM  =  Index of word found in dictionary list
0045      *               IVAL    =  Integer value if WHAT=2
0046      *               VALUE   =  Real value if WHAT=3
0047      *
0048            COMMON /INPUT/  IMAGE,NEXTRD,IC,FCHAR,LCHAR,WHAT,LSTPOS,
0049           &                WRDNUM,IVAL,VALUE
0050
0051            VALUE=0.
0052            IVAL=0
0053            WRDNUM=0
0054            LASTC=LLEN(IMAGE)
0055            SQUOTE=''''
0056            DELIM=' ,       '       !<space>,<comma>,<tab>
0057
```

```
0058              IF(NEXTRD.EQ.0) THEN
0059     !          PRINT *,'IC: ',IC,'  LC: ',LASTC
0060     !        print*,image(ic:lastc)
0061               IF(IC.LE.LASTC) GOTO 10
0062               WHAT=1
0063               RETURN
0064              ELSE IF(NEXTRD.EQ.1) THEN
0065     !          CALL STRINP(IMAGE,LASTC)
0066               LASTC=GET_STRING(IMAGE)
0067               IF(LASTC.EQ.0) THEN
0068                 WHAT=1
0069                 RETURN
0070               ELSE IF(LASTC.EQ.-2) THEN
0071                 WHAT=6
0072                 RETURN
0073               ENDIF
0074     !        print*,image(1:lastc)
0075               CALL CTB(IMAGE)          ! change commas to spaces
0076               CALL TTB(IMAGE)          ! change <tab>s to spaces
0077               CALL CMPRES(IMAGE)       ! convert multiple spaces to one space
0078               LASTC=LLEN(IMAGE)        ! (possibly) new length of string
0079     !        print*,image(1:lastc)
0080               IC=1
0081              ELSE IF(NEXTRD.EQ.2) THEN
0082               IC=1
0083              ENDIF
0084
0085        10    IF(INDEX(DELIM,IMAGE(IC:IC)).NE.0) THEN
0086               IC=IC+1
0087               GOTO 10
0088              ENDIF
0089     !        PRINT *,'AFTER 10$ IC:  ',IC,' LASTC: ',LASTC
0090              IF(IC.GT.LASTC) THEN
0091               WHAT=1
0092               RETURN
0093              ENDIF
0094              IF(IMAGE(IC:IC).EQ.SQUOTE) THEN
0095               WHAT=7
0096               LSTPOS=IC
0097               FCHAR=IC+1
0098               LCHAR=INDEX(IMAGE(FCHAR:),SQUOTE)
0099               IF(LCHAR.NE.0) GOTO 20
0100               WRITE(6,1000) IMAGE(1:LLEN(IMAGE))
0101      1000     FORMAT(1X,'End of Record within string (DCIFER):'/,1X,A)
0102               WHAT=1
0103               RETURN
0104        20     IC=LCHAR+1
0105               LCHAR=LCHAR+FCHAR-2
0106               RETURN
0107              ELSE IF(ISDIG(IMAGE(IC:IC)).OR.
0108           &                    (INDEX('+-',IMAGE(IC:IC)).NE.0)) THEN
0109               IFDP=.FALSE.
0110               DO IB=IC,LASTC
0111                 IF(INDEX('0123456789+-. ',IMAGE(IB:IB)).EQ.0) GOTO 30
0112                 IFDP=(IFDP.OR.(IMAGE(IB:IB).EQ.'.'))
0113               END DO
0114               IB=LASTC+1
```

```
0115        30      IEND=IB-1
0116                IF(IFDP) THEN
0117                  WHAT=3
0118    !           print *,image(ic:iend)
0119                  VALUE=TOREAL2(IMAGE(IC:IEND))
0120                  IF(VALUE.EQ.-9999999.) GOTO 60
0121                ELSE
0122                  WHAT=2
0123    !           print *,'DCIFER: IC:',ic,'  IEND:',iend
0124                  IVAL=TOINT(IMAGE(IC:IEND))
0125                  IF(IVAL.EQ.-214 83648) GOTO 60
0126                ENDIF
0127                IC=IEND+1
0128                RETURN
0129              ELSE
0130                LSTPOS=IC
0131                DO ID=IC,LASTC
0132                  IF(INDEX(DELIM,IMAGE(ID:ID)).NE.0) GOTO 40
0133                END DO
0134                ID=LASTC
0135        40      WORD=IMAGE(IC:ID)
0136    !           PRINT *,'At 50$, WORD=',word
0137                DO IE=1,NDICT
0138                  IF(WORD(1:NCLST).EQ.LIST(IE)) THEN
0139                    WRDNUM=IE
0140                    WHAT=5
0141                    GOTO 50
0142                  ENDIF
0143                END DO
0144                WHAT=4
0145              ENDIF
0146        50    IC=ID+1
0147              RETURN
0148        60    WRITE(6,1010)
0149      1010    FORMAT(1X,'Error in numeric field (DCIFER)')
0150              END
```

SUBROUTINE DIRECT

```
0001    ************************************************************************
0002    ************************************************************************
0003            SUBROUTINE DIRECT
0004    *  DIRECT:
0005    *       Directory of SCRTCH.DAT for XTRAC and DSPLAY
0006    *
0007            INCLUDE 'XTRBLK/LIST'
0008
0009            READ(1,REC=1,ERR=10) NVAR,NAME,RUN,MAX,MIN,UNITS,NUMB
0010            IF(NVAR.EQ.0) GOTO 10
0011            DO I=1,NVAR
0012              WRITE(6,1000) I,RUN(I),NAME(I),MIN(I),MAX(I),NUMB(I)
0013     1000     FORMAT(1X,I3,')',2X,2A10,2(2X,G),I5)
0014            END DO
0015            RETURN
0016       10   WRITE(6,1010)
0017     1010   FORMAT(/1X,'Empty'/)
0018            RETURN
0019            END
```

SUBROUTINE STANDEV

```
0001    ************************************************************************
0002            subroutine standev(filnam,varnam)
0003    *                                                               R. Stevens
0004    *       standev is called by xtrac as part of the               12/18/84
0005    *       STA command for the Spenny Neck Analysis                SDC
0006    *       package. standev opens the scratch file and finds all
0007    *       occurances of the passed varnam and saves the
0008    *       corresponding record number in recnumr. standev then calls
0009    *       stdcal to get values for the mean and standard deviation
0010    *       at each timestep for the set of varnam. two variables are
0011    *       then calculated, sigup; mean plus st. dev. at each timestep,
0012    *     - and siglo; mean minus st.dev. at each timestep. The scratch
0013    *       file is then rewritten.
0014    *
0015            real mean(600),sigup(600),siglo(600),stdev(600)
0016            character filnam*30,varnam*6
0017            integer varcount,recnumr(100)
0018            include 'xtrblk/list'
0019    *
0020            open(unit=1,file=filnam,status='old',access='direct',
0021         &  organization='relative',iostat=ios,err=100)
0022    *
0023            read(1,rec=1,iostat=ios,err=110)nvar,name,run,max,
0024         &   min,units,numb
0025    *
0026            j=0
0027            minpts=600                              ! 600 is > we need
0028            varcount=0
0029            do i=1,nvar
0030                if(name(i).eq.varnam)then
0031                   varcount=varcount+1              ! number of variables
0032                   j=j+1
0033                   recnumr(j)=i+1                   ! keep track of records
0034                   if(numb(i).lt.minpts) minpts=numb(i)
0035                endif
0036            enddo
0037    *
0038            if(varcount.eq.0)then     ! make sure we have enough vars
0039                write(6,2001)varnam
0040     2001       format(1x,'The variable ',a,' is not in the scratch file.'/)
0041                write(6,2010)
0042     2010       format(1x,'Please try again...'/)
0043                return
0044            elseif(varcount.eq.1)then
0045                write(6,2002)varnam
0046     2002       format(1x,'Variable ',a,' occurs only once in the scratch
0047         &  file.'/1x,'STAing it wont prove anything.'/)
0048                write(6,2010)
0049                return
0050            endif
0051    *
0052            call stdcal(recnumr,minpts,varcount,filnam,stdev,mean)
0053    *
0054            do i=1,minpts
0055                sigup(i)=mean(i) + stdev(i)
0056                siglo(i)=mean(i) - stdev(i)
0057            enddo
```

```
0058    *
0059            meanrec=nvar+2          ! prepare info for scrtch file
0060            siguprec=nvar+3
0061            siglorec=nvar+4
0062            nvar1=nvar+1
0063            nvar2=nvar+2
0064            nvar3=nvar+3
0065            numb(nvar1)=minpts
0066            numb(nvar2)=minpts
0067            numb(nvar3)=minpts
0068            run(nvar1)='      '
0069            run(nvar2)='      '
0070            run(nvar3)='      '
0071            units(nvar1)=0
0072            units(nvar2)=0
0073            units(nvar3)=0
0074            call mnmx(mean,min,max,nvar1,minpts)
0075            call mnmx(sigup,min,max,nvar2,minpts)
0076            call mnmx(siglo,min,max,nvar3,minpts)
0077            name(nvar1)='MO'//varnam(1:3)//varnam(6:6)
0078            name(nvar2)='MP'//varnam(1:3)//varnam(6:6)
0079            name(nvar3)='MM'//varnam(1:3)//varnam(6:6)
0080            nvar=nvar + 3
0081                                                    ! fix scrtch file
0082            write(1,rec=1) nvar,name,run,max,min,units,numb
0083            write(1,rec=meanrec) (mean(i),i=1,numb(nvar1))
0084            write(1,rec=siguprec) (sigup(i),i=1,numb(nvar2))
0085            write(1,rec=siglorec) (siglo(i),i=1,numb(nvar3))
0086    *
0087            close(unit=1)
0088    *
0089            return
0090     100    continue
0091            print *,'open error'
0092            return
0093     110    continue
0094            print *,'read error'
0095            return
0096            end
0097
```

SUBROUTINE STDCAL

```
0001    ************************************************************************
0002            subroutine stdcal(recnumr,minpts,varcnt,filnam,stdev,mean)
0003    *
0004    *       stdcal is called by standev as part of the      R. Stevens
0005    *       STA command of the Spenny Neck Analysis         12/18/84
0006    *       package. stdcal allocates virtual memory        SDC
0007    *       and then loads desired arrays (rec nums for desired arrays
0008    *       are passed in recnumr) into one large array in virtual
0009    *       memory. this array is passed to smean, which does the
0010    *       calculations.
0011    *
0012            character filnam*30
0013            integer minpts,varcnt,recnumr(varcnt),addr1,addr2
0014            real stdev(minpts),array(600),mean(minpts)
0015    *
0016            nbytes=minpts*varcnt*4                  ! size of total vm
0017            nqw=nbytes/8
0018            if(mod(nbytes,8).ne.0)nqw=nqw+1
0019            nvb=nqw*8
0020    *
0021            kstat=lib$get_vm(nvb,addr1)             ! allocate vm
0022            if(.not.kstat) call lib$stop(%val(kstat))
0023    *
0024            nbytes2=minpts*4                        ! size of each array
0025            addr2=addr1
0026    *
0027            do i=1,varcnt                           ! read in arrays
0028                read(1,rec=recnumr(i),iostat=ios,err=100)
0029         &      (array(j),j=1,minpts)
0030                call lib$movc3(minpts*4,array(1),%val(addr2))
0031                addr2=addr2+nbytes2
0032            enddo
0033    *
0034            call smean(%val(addr1),minpts,varcnt,stdev,mean)
0035    *
0036            kstat=lib$free_vm(nvb,addr1)
0037            if(.not.kstat) call lib$signal(%val(kstat))
0038    *
0039            return
0040     100    continue
0041            write(6,101)
0042     101    format(1x,'READ ERROR in module STDCAL, call a programmer.')
0043            return
0044            end
```

SUBROUTINE SMEAN

```
      subroutine smean(x,npts,nvars,sdev,mean)
*
*     smean calculates the mean and
*     standard deviation for the STA
*     command of the Spenny Neck Analysis
*     Package.  nvars arrays, each npts long ~are packed
*     into x by subroutine stdcal. The mean and std. dev.
*     are calculated for the set of arrays at each timestep.
*
      real x(npts*nvars),sdev(npts),mean(npts)
*
*
      do i=1,npts                          !calc mean
            kount=i
            xbar=0.0
            do j=1,nvars
                  xbar=xbar + x(kount) ! sum values
                  kount=kount + npts
            enddo
            mean(i)=xbar/nvars
      enddo
*
      do i=1,npts                          ! calc std. dev.
            kount=i
            sum=0.0
            do j=1,nvars
                  sum=sum+(x(kount)-mean(i))**2
                  kount=kount+npts
            enddo
            sdev(i)=sqrt(sum/(nvars-1))
      enddo
*
      return
      end
```

SUBROUTINE MATH

```
0001    ************************************************************************
0002    ************************************************************************
0003            SUBROUTINE MATH(ICMD)
0004    *  MATH:
0005    *       Performs the 'VMA', 'ADD', 'SUB', 'CON', 'DIV',
0006    *       and 'NOR' commands for XTRAC
0007    *
0008            BYTE OK
0009            INTEGER KEEP(3),SIGLEN,WKEEP(3)
0010            REAL A(3,598)
0011            CHARACTER RSAVE*6,VSAVE*6,DUMMY*6,NEWVAR*6
0012
0013            INCLUDE 'XTRBLK/LIST'
0014    *
0015    *       Read in the dictionary
0016    *
0017            READ(1,REC=1,ERR=2010) NVAR,NAME,RUN,MAX,MIN,UNITS,NUMB
0018    !       do i=1,nvar
0019    !         print *,name(i)
0020    !       enddo
0021    *
0022    *       Keyword RUN
0023    *
0024            CALL DCIFER(3,1,LIST2)
0025            IF(WHAT.NE.5) THEN
0026              WRITE(6,2000)
0027     2000     FORMAT(1X,'The word RUN must follow the command')
0028              WRITE(6,1010)
0029              RETURN
0030            ENDIF
0031    *
0032    *       Run number
0033    *
0034            CALL DCIFER(6,380,LISTR)
0035            IF(WHAT.NE.5) THEN
0036              WRITE(6,1000)
0037     1000     FORMAT(1X,'No such Run Number')
0038              WRITE(6,1010)
0039     1010     FORMAT(1X,'Please re-enter complete line')
0040              RETURN
0041            ENDIF
0042    !       print *,'wrdnum=',wrdnum
0043            RSAVE=LISTR(WRDNUM)
0044    !       print *,'rsave=',rsave
0045    *
0046    *       Keyword VAR
0047    *
0048            CALL DCIFER(3,1,'VAR')
0049            IF(WHAT.NE.5) THEN
0050              WRITE(6,1020)
0051     1020     FORMAT(1X,'Keyword VAR is missing')
0052              WRITE(6,1010)
0053              RETURN
0054            ENDIF
0055    *
0056    *       Variable name(s)
0057    *
```

```
0058            IF(ICMD.GT.3) THEN
0059              IVEND=1
0060            ELSE IF(ICMD.EQ.1) THEN
0061              IVEND=3
0062            ELSE
0063              IVEND=2
0064            ENDIF
0065            DO IVKNT=1,IVEND
0066              CALL DCIFER(6,NVAR,NAME)
0067              IF(WHAT.NE.5) THEN
0068            print *,'what =',what
0069                WRITE(6,1030)
0070     1030       FORMAT(1X,'No such variable'/)
0071                WRITE(6,1010)
0072                RETURN
0073              ENDIF
0074              VSAVE=NAME(WRDNUM)
0075            print *,'vsave=',vsave
0076              DO IA=1,NVAR
0077                IF(RUN(IA).EQ.RSAVE.AND.NAME(IA).EQ.VSAVE) GOTO 10
0078              END DO
0079              WRITE(6,1040) RSAVE,VSAVE
0080     1040     FORMAT(1X,'Run ',A,' variable ',A,' has not been EXTracted'/)
0081              WRITE(6,1010)
0082              RETURN
0083       10     KEEP(IVKNT)=IA+1
0084              WKEEP(IVKNT)=NUMB(IA)
0085            END DO
0086            IF(ICMD.EQ.6) GOTO 20
0087            IF(ICMD.GT.3) THEN
0088              CALL DCIFER(6,1,DUMMY)
0089              IF(WHAT.NE.3) THEN
0090                WRITE(6,1050)
0091     1050       FORMAT(1X,'Real constant is missing'/)
0092                WRITE(6,1010)
0093                RETURN
0094              ENDIF
0095              RCONST=VALUE
0096              IF(RCONST.EQ.0.0.AND.ICMD.EQ.5) THEN
0097                WRITE(6,1060)
0098     1060       FORMAT(1X,'Real constant is zero - DIVIDE ignored'/)
0099                RETURN
0100              ENDIF
0101            ELSE
0102              OK=.TRUE.
0103              DO IZ=1,IVEND-1
0104                OK=(WKEEP(IZ).EQ.WKEEP(IZ+1)).AND.OK
0105              END DO
0106              IF(.NOT.OK) THEN
0107                WRITE(6,1070)
0108     1070       FORMAT(1X,'It is not possible to combine sensor and',
0109         &        ' photo variables in this command.'/)
0110                WRITE(6,1010)
0111                RETURN
0112              ENDIF
0113            ENDIF
0114     *
```

```
0115    *       Get 'Use Variable'
0116    *
0117       20   CALL DCIFER(6,1,DUMMY)
0118            IF(WHAT.NE.7) THEN
0119    !       print *,'What =',what
0120              WRITE(6,1080)
0121     1080     FORMAT(1X,'The "use variable" must begin with an alphabetic',
0122         &           /1X,'and be enclosed in single quotes'/)
0123              WRITE(6,1010)
0124              RETURN
0125            ENDIF
0126    !       print *,fchar,lchar
0127            NEWVAR=IMAGE(FCHAR:LCHAR)
0128    !       print *,newvar
0129    *
0130    *       Read 'em in
0131    *
0132            JNPTS=NUMB(KEEP(1)-1)
0133    !       print *,'jnpts=',jnpts
0134            DO IB=1,IVEND
0135              READ(1,REC=KEEP(IB)) (A(IB,II),II=1,JNPTS)
0136            END DO
0137            IF(ICMD.EQ.6) THEN
0138              RCONST=A(1,1)
0139              IF(RCONST.EQ.0.0) THEN
0140                WRITE(6,1090)
0141     1090       FORMAT(1X,'Initial value zero - NORMALIZE ignored.'/)
0142                RETURN
0143              ENDIF
0144            ENDIF
0145            DO 70 IDK=1,JNPTS
0146    !       print *,'icmd=',icmd
0147              GOTO (30,40,50,60,60),ICMD-1
0148    *   'VMA'
0149            E(IDK)=SQRT(A(1,IDK)**2+A(2,IDK)**2+A(3,IDK)**2)
0150            GOTO 70
0151    *   'ADD'
0152       30   E(IDK)=A(1,IDK)+A(2,IDK)
0153            GOTO 70
0154    *   'SUB'
0155       40   E(IDK)=A(1,IDK)-A(2,IDK)
0156            GOTO 70
0157    *   'CON'
0158       50   E(IDK)=A(1,IDK)+RCONST
0159            GOTO 70
0160    *   'DIV' & 'NOR'
0161       60   E(IDK)=A(1,IDK)/RCONST
0162       70   CONTINUE
0163    *
0164    *       Now write it out
0165    *
0166            NVAR=NVAR+1
0167            RUN(NVAR)=RSAVE
0168            NAME(NVAR)=NEWVAR
0169            UNITS(NVAR)=UNITS(KEEP(1)-1)
0170            NUMB(NVAR)=WKEEP(1)
0171            CALL MNMX(E,MIN,MAX,NVAR,JNPTS)
```

```
0172              WRITE(1,REC=1) NVAR,NAME,RUN,MAX,MIN,UNITS,NUMB
0173              IQZX=NVAR+1
0174              WRITE(1,REC=IQZX) (E(II),II=1,JNPTS)
0175              RETURN
0176      *
0177      *       Error exit
0178      *
0179       2010   WRITE(6,2020)
0180       2020   FORMAT(1X,'No Variables EXTRACTed -- command ignored'//)
0181              RETURN
0182              END
```

SUBROUTINE MNMX

```
0001    ************************************************************************
0002    ************************************************************************
0003            SUBROUTINE MNMX(ARR,MIN,MAX,NVAR,NUMMER)
0004    *  MNMX:
0005    *       Find the min & max values of ARR and insert in MIN(NVAR)
0006    *       and MAX(NVAR)
0007    *
0008            REAL MIN(100),MAX(100),ARR(801)
0009            QNTMP=99999.
0010            QXTMP=-99999.
0011            DO I=1,NUMMER
0012              IF(ARR(I).LT.QNTMP) QNTMP=ARR(I)
0013              IF(ARR(I).GT.QXTMP) QXTMP=ARR(I)
0014            END DO
0015            print*,'MNMX:   min                max                nvar'
0016            print *,qntmp,qxtmp,nvar
0017            MIN(NVAR)=QNTMP
0018            MAX(NVAR)=QXTMP
0019            RETURN
0020            END
```

SUBROUTINE CURD

```
0001    ************************************************************************
0002            subroutine curd(xarr,yarr,npts,isymb)
0003    *  CURD:                                                  Rick Stevens
0004    *                                                         SDC
0005    *                                                         Aug  7,84
0006    *
0007    *       From a routine by Doug Gordon:
0008    *        Routine to simulate the DISSPLA CURVE routine.
0009    *        XARR is x-axis values, YARR is y-axis values. NPTS is
0010    *        number of points to plot. ISYMB is the symbol to use
0011    *        at each point. If ISYMB is > 0, symbols are drawn.
0012    *        If ISYMB = 0, a solid line is drawn. If ISYMB < 0,
0013    *        dashed lines are drawn. Currently, 5 line types
0014    *        are supported.
0015    *
0016            byte outside
0017            real xarr(npts),yarr(npts)
0018            integer jsymb(5)                              ! line types
0019            data jsymb/54,32,312,5434,74/
0020
0021            shgt=0.08                   !Symbol height
0022            ksymk=1                     !determine frequency of symbols
0023            if(npts.ge.50) ksymk=npts/10
0024            if(npts.ge.1000) ksymk=npts/100
0025            i=0
0026            outside=.true.
0027            call seedw(vxn,vxx,vyn,vyx)                   ! see virtual window
0028
0029            do while(outside)           !while inside virtual window
0030              i=i+1
0031              if(i.gt.npts) return
0032              x=xarr(i)
0033              y=yarr(i)
0034              outside=(x.lt.vxn .or.  x.gt.vxx .or.       ! Test that point lies
0035         &       y.lt.vyn .or.  y.gt.vyx)                 ! in virtual window
0036            end do
0037
0038            call movea(xarr(i),yarr(i))
0039            if(isymb.gt.0) call symbol(xarr(i),yarr(i),isymb,shgt)
0040            isk=0
0041            do j=i,npts
0042             if(isymb.gt.0)then                           ! want symbol
0043              call drawa(xarr(j),yarr(j))
0044              if(j.eq.npts) isk=ksymk       ! always put sym on last pt
0045              if(isymb.gt.0.and.isk.ge.ksymk) then
0046                call symbol(xarr(j),yarr(j),isymb,shgt)
0047                isk=0                                    ! symbol counter
0048              endif
0049             else if(isymb.eq.0)then                      ! want plain old line
0050                call drawa(xarr(j),yarr(j))
0051             else                                         ! want dashed line
0052              k=-isymb
0053              jazz=jsymb(k)
0054              call dasha(xarr(j),yarr(j),jazz)
0055             endif
0056              isk=isk+1
0057            end do
```

```
0058            call seetw(xm, xx, ym, yx)              ! see screen window
0059            call movabs(xm, yx)                     ! place cursor
0060            call tsend                              ! dump output buffer
0061            return
0062            end
```

SUBROUTINE LABLE

```
**********
**********
      subroutine lable(rntmp,read1,read2,quit,lintyp)

      byte quit
      real xarr(2),yarr(2)
      integer kount,ltype(20)
      character*6 plots(20),rntmp
      character read1*6,read2*6
      character str*24
      character*6 r1(20),r2(20)
      data kount/0/

      save kount

      include 'xtrblk.for/list'
      include 'dsppltblk.for/list'

      if(.not.quit)then
        kount=kount+1                          ! Note: kount retains its!
        r1(kount)=read1                        ! value when lable is!
        r2(kount)=read2                        ! exited and has that!
        plots(kount)=rntmp                     ! value upon return to lable!
        ltype(kount)=lintyp
        return
      endif

      call seetw(minx,maxx,miny,maxy)
      nminx=maxx
      nmaxx=1023
      call csize(ihorz,ivert)

      call twindo(nminx,nmaxx,miny,maxy)
      call dwindo(0.,360.,0.,200.)
      yy=200.
      do i=1,kount
        xx=30.
        yy=yy-10.
        call movea(xx,yy)
        str=plots(i)//' '//(r1(i)(1:llen(r1(i))))//' vs '//r2(i)
        cxx=xx+(llen(str)*ihorz)
        cyy=yy-(ivert/5)
        xarr(1)=xx
        xarr(2)=cxx
        yarr(1)=cyy
        yarr(2)=cyy
        call tekout(str)
      enddo
      kount=0
      idfl = llen(dsplay$figure)
      if(idfl .gt. 0) then
        call movabs(nmaxx-(ihorz*(idfl+5)), miny+2)
        call tekout('Fig. '//dsplay$figure(1:idfl))
        call tsend
      endif

      return
      end
```

SUBROUTINE SYMBOL

```
0001            subroutine symbol(x,y,isym,sizin)
0002      *  SYMBOL:                                    Douglas A  Gordon
0003      *                                             Arcon Corporation
0004      *                                             3-APR-1984
0005      *
0006      *       Tektronix routine to generate a symbol at the point
0007      *       (X,Y) [assumes a virtual window was declared].
0008      *       ISYM is the symbol number. SIZIN is the symbol size
0009      *       in inches.  Note that all symbols except the triangles are
0010      *       affected by rotation of the plot.
0011      *
0012      *       Modified 10-MAY-1984.  Added 3 new symbols
0013      *
0014            call seedw(vxn,vxx,vyn,vyx)
0015            if(x.lt.vxn .or.  x.gt.vxx .or.          ! Test that point lies
0016           &     y.lt.vyn .or.  y.gt.vyx) return     ! in virtual window
0017            call movea(x,y)          ! move to point in virtual coords
0018            call seeloc(ixa,iya)     ! get the position in screen units
0019            ksiz=kin(sizin)
0020            irad=ksiz/2
0021            rad=float(irad)
0022            sq32=sqrt(3.0)/2.0
0023
0024            goto (10,20,30,40,50,60,70),isym-1      !default is circle
0025      *
0026      * 1)    Circle centered on (X,Y) radius = IRAD
0027      *
0028            call movrel(irad,0)
0029            do ang= 10.0, 360., 10.0
0030              ixp=ixa+int(rad*sind(ang))
0031              iyp=iya+int(rad*cosd(ang))
0032              call drwabs(ixp,iyp)
0033            end do
0034            goto 1000
0035      *
0036      * 2)    Square centered on (X,Y) , length of side = KSIZ
0037      *
0038         10   call movrel(irad,-irad)
0039              call drwrel(0,ksiz)
0040              call drwrel(-ksiz,0)
0041              call drwrel(0,-ksiz)
0042              call drwrel(ksiz,0)
0043              goto 1000
0044      *
0045      * 3)    Plus sign centered on (X,Y)
0046      *
0047         20   call movrel(-irad,0)
0048              call drwrel(ksiz,0)
0049              call movrel(-irad,-irad)
0050              call drwrel(0,ksiz)
0051              goto 1000
0052      *
0053      * 4)    Equilateral triangle centered on (X,Y), altitude = KSIZ
0054      *
0055         30   call movrel(0,irad)              ! apex of triangle
0056              ixp=ixa-int(rad*sq32)
0057              iyp=iya-irad
```

```
0058            call drwabs(ixp,iyp)
0059            ixp=ixa+int(rad*sq32)
0060            call drwabs(ixp,iyp)
0061            call drwabs(ixa,iya+irad)
0062            goto 1000
0063    *
0064    * 5)    X centered on (X,Y) bounded by square with sides = KSIZ
0065    *
0066       40   call movrel(-irad,irad)
0067            call drwrel(ksiz,-ksiz)
0068            call movrel(-ksiz,0)
0069            call drwrel(ksiz,ksiz)
0070            goto 1000
0071    *
0072    * 6)    Asterisk centered on (X,Y) bounded by square sides = KSIZ
0073    *
0074       50   call movrel(-irad,0)
0075            call drwrel(ksiz,0)
0076            call movrel(-irad,-irad)
0077            call drwrel(0,ksiz)
0078            call movea(x,y)
0079            call movrel(-irad,irad)
0080            call drwrel(ksiz,-ksiz)
0081            call movrel(-ksiz,0)
0082            call drwrel(ksiz,ksiz)
0083            goto 1000
0084    *
0085    * 7)    Upside-down triangle
0086    *
0087       60   call movrel(0,-irad)
0088            ixp=ixa-int(rad*sq32)
0089            iyp=iya+irad
0090            call drwabs(ixp,iyp)
0091            ixp=ixa+(rad+sq32)
0092            call drwabs(ixp,iyp)
0093            call drwabs(ixa,iya-irad)
0094            goto 1000
0095    *
0096    * 8)    Diamond
0097    *
0098       70   call movrel(-irad,0)
0099            call drwrel(irad,irad)
0100            call drwrel(irad,-irad)
0101            call drwrel(-irad,-irad)
0102            call drwrel(-irad,irad)
0103            goto 1000
0104    *
0105    *       Insert additional figures here
0106    *
0107
0108     1000   call movea(x,y)          ! restore original position
0109            return
0110            end
```

SUBROUTINE TITLE

```
0001    ************************************************************************
0002    ************************************************************************
0003            subroutine title(ttl,lttl,xlabl,lx,ylabl,ly,dum1,dum2)
0004    *  TITLE:                                    Douglas A. Gordon
0005    *                                            Arcon Corporation
0006    *                                            08-FEB-1984
0007    *
0008    *       Routine to emulate the DISSPLA  TITLE routine using
0009    *       tektronix  PLOT-10.     DUM1  & DUM2  are  the  axis
0010    *       lengths in DISSPLA, but aren't needed here
0011    *
0012    *     - Modified 09-MAY-1984 to skip over  titles or labels
0013    *         passed with a length of zero or blank strings
0014    *
0015    *       Modified  20-Jul-1984   to  use   TEKOUT,  and   to
0016    *         eliminate integer storage arrays for strings.
0017    *
0018            integer ychar
0019            character ttl*(*),xlabl*(*),ylabl*(*)
0020
0021            call seetw(ixn,ixx,iyn,iyx)
0022            lh1=linhgt(2)
0023            lw1=linwdt(0)
0024            if(ttl.eq.' ') lttl=0
0025            if(xlabl.eq.' ') lx=0
0026            if(ylabl.eq.' ') ly=0
0027            if(lttl.gt.0) call centre(ttl)
0028            if(lx.gt.0) call centre(xlabl)
0029            if(ly.gt.0) call centre(ylabl)
0030
0031            if(lttl.gt.0) then
0032              ixpos=ixn+(ixx-ixn-linwdt(len(ttl)))/2
0033              iypos=iyx+(780-iyx)/2
0034              call movabs(ixpos,iypos)
0035              call tekout(ttl)
0036            endif
0037
0038            if(lx.gt.0) then
0039              ixpos=ixn+(ixx-ixn-linwdt(len(xlabl)))/2
0040              call movabs(ixpos,linhgt(1)/2)
0041              call tekout(xlabl)
0042            endif
0043
0044            if(ly.gt.0) then
0045              ilh=linhgt(1)
0046              iypos=iyx-(iyx-iyn-linhgt(len(ylabl)))/2
0047              do i=1,llen(ylabl)
0048                ychar=ichar(ylabl(i:i))
0049                call movabs(lw1,iypos)
0050                call ancho(ychar)
0051                iypos=iypos-ilh
0052              end do
0053            endif
0054
0055            call tsend
0056            return
0057            end
```

SUBROUTINE DSP-WPAGE

```
**********************************************************************
**********************************************************************
        subroutine dsp_wpage(xsize,ysize,xlen,ylen)
*  WPAGE:                                                Douglas A. Gordon
*                                                        Arcon Corporation
*                                                        08-FEB-1984
*
*       Routine to establish the logical terminal window for
*       a tektronix terminal.  Attempts to somewhat duplicate
*       the DISSPLA PAGE routine.  XSIZE is x page length in
*       inches, YSIZE is the y page length in inches.  XLEN is
*       the x-axis length and YLEN is the y-axis length
*
        parameter(maxx=1024)
        parameter(maxy=780)

        ixpts=int(xlen/xsize*float(maxx))
        iypts=int(ylen/ysize*float(maxy))

        ixn=150                    ! mod by richs!
        ixx=ixn+ixpts
        iyn=(maxy-iypts)/2
        iyx=iyn+iypts

        call twindo(ixn,ixx,iyn,iyx)
        return
        end
```

SUBROUTINE DSP-OVERLAY

```
************************************************************************
************************************************************************
      Subroutine Dsp_Overlay(X, Y, Npts, Ncurv, Maxpts, Isym,
     &                                   Ttl, Xlabl, Ylabl, Flags)
*
*  OVERLAY:                            Douglas A. Gordon
*                                      Arcon Corporation
*                                      16-Apr-1985
*      Created from OMNIPLOT.FOR
*
*  Last Revision Date: Thu  5-Dec-85 14:07
*
*  Abstract:
*
*      Simple overlay plotting routine.
*
*  Calling Sequence:
*
*      CALL OVERLAY( X.rf.ra, Y.rf.ra, NPTS.rl.r, NCURV.rl.r,
*              MAXPTS.rl.r, ISYM.rl.r, TTL.rt.dx, XLABL.rt.dx,
*              YLABL.rt.dx, FLAGS.rl.r)
*
*  Formal Parameters:
*
*      X       Two-dimensional array (dimensioned (MAXPTS,NCURV))
*                of X-coordinates for plotting.  Passed by
*                reference.
*      Y       Two-dimensional array (dimensioned (MAXPTS,NCURV))
*                of Y-coordinates for plotting.  Passed by
*                reference.
*      NPTS    Array (dimensioned (NCURV)) containing the number
*                of points to plot for the corresponding X and Y
*                arrays.  Integer*4.  Passed by reference.
*      NCURV   Number of curves to plot.  Used as the implied
*                dimension of X, Y, and NPTS. Integer*4. Passed
*                by reference.
*      MAXPTS  The upper dimension for the number of points in
*                the arrays X and Y.  Integer*4. Passed by reference.
*      ISYM    The starting symbol value.  Ignored if bit 8 of FLAGS
*                set. Set to one if zero or negative and bit 8 clear.
*                Integer*4.  Passed by reference.
*      TTL     Title for plot.  Passed length character
*                string.  Passed by descriptor.
*      XLABL   Label for X-axis.  Passed length character
*                string.  Passed by descriptor.
*      YLABL   Label for Y-axis.  Passed length character
*                string.  Passed by descriptor.
*      FLAGS   Plot customization flags. Integer*4. The
*                following bits are defined:
*
*                Bit     Meaning if set                  Value
*                ---     ------------------------------  -----
*                 0      Draw box axes                       1
*                 1      Draw line at Y=0.0 (virtual)        2
*                 2      Draw line at X=0.0 (virtual)        4
```

```
*                3       Draw a point grid on the plot          8
*                4       Auto Hardcopy plot                    16
*                5       Invoke VT240/241 for duration
*                          of plot - exit to VT100 mode        32
*                6       X-axis scale from common              64
*                7       Y-axis scale from common             128
*                8       Suppress symbols                     256
*                9       Dashed lines (not currently
*                          supported)                         512
*               10       Used by the legend software       1024
*               11       Rotate auto-hardcopy (VT240 with
*                          V2.1 firmware or later)          2048
*               12       Draw a scatter plot rather than
*                          a curve                          4096
*               13       Laser printer support            8192
*               14       Enable message trapping         16384
*               15-31    Undefined (must be zero)
*
*  Implicit Inputs:
*
*       None.
*
*  Implicit Outputs:
*
*       Plot to the terminal.
*
*  Side Effects:
*
*       Plays with the emulations settings on VT240 series terminals.
*
*  Functions & Subroutines Called:
*
*                AXES1             TSC Plot Library
*       I*4      BAUDQ             TSC Plot Library
*       I*4      GET_ARRAY         TSC General Library
*                HARD_COPY         TSC Plot Library
*                HARD_COPY_FF      TSC Plot Library
*                INITT             TEKTRONIX Terminal Control System
*                LIB$SIGNAL        VAX Run Time Library
*       I*4      LLEN              TSC General Library (MACRO32)
*                MTITLE            TSC Plot Library
*                NEWPAG            TEKTRONIX Terminal Control System
*                N_CURVE           TSC Plot Library
*                N_SPLATTER        TSC Plot Library
*       I*4      PARSE_BIT_FLAGS   TSC General Library (MACRO32)
*                PLHOLD            TSC Plot Library
*       I*4      PUTMSG            TSC General Library (MACRO32)
*                QUIET_PLOT        TSC Plot Library
*                SCAL              TSC Plot Library
*                VT200_SET_MODE    TSC General Library
*                WPAGE             TSC Plot Library
*
*  Revision History:
*
*       Some revision history removed on conversion.
```

```
*
*       Modified 10-Jun-1985 to support additional functionality in VT240's
*         with firmware upgrade.
*       Modified 11-Jun-1985.  Moved the rotate code since VT240 doesn't
*         recognize escape sequences in Tek emulation.
*       Modified 28-Jun-1985.  Changed AXES_MASK to pass additional bits.
*       Modified  9-Jul-1985 to support new hardcopy software.
*       Modified 23-Jul-1985.  Added call to RECOLOR.
*       Modified 10-Sep-1985 to return VT240's to VT240 7-bit controls
*         rather than VT100 emulation (VMS V4.1 upgrade)
*       Modified 19-Sep-1985 to allow scatter plots.
*       Modified 15-Nov-1985 to include a stab in the dark to support the
*         laser printer.
*
        byte first/.true./, autohc, vt240, bit_values(0:31), manualx,
     &    manualy, no_sym, rotate, scatter, laser
        integer*4 ncurv, npts(ncurv), flags, max_bit, status, axes_flags,
     &    axes_mask/'0000003F'X/, lttl, lx, ly, jsym, maxpts
        integer*4 parse_bit_flags, baudq, llen, get_array, putmsg
        real x(maxpts,ncurv), y(maxpts,ncurv), xmin, xmax, ymin, ymax, tmp(2)
        character*(*) ttl, xlabl, ylabl, esc*1/27/

        include 'pltdef.for/list'
        include 'pltmsgdef.for/list'            ! 40 lines
        include 'dsppltblk.for/list'            ! 5 lines

        parameter (max_bit = plt$max_bit)

        equivalence (bit_values(plt$v_autohc), autohc)
        equivalence (bit_values(plt$v_vt240), vt240)
        equivalence (bit_values(plt$v_xscale), manualx)
        equivalence (bit_values(plt$v_yscale), manualy)
        equivalence (bit_values(plt$v_nosym), no_sym)
        equivalence (bit_values(plt$v_rotate), rotate)
        equivalence (bit_values(plt$v_scatter), scatter)
        equivalence (bit_values(plt$v_laser), laser)
*
*       Test for reasonable input
*
        do i = 1, ncurv
          if(npts(i) .lt. 0) then
            status = putmsg(%val(plt$_negnumpts), %val(1), %val(i))
            if(.not. status) call lib$signal(%val(status))
            return
          endif
        end do
*
*       Parse the option bits
*
        status = parse_bit_flags(4, flags, max_bit, bit_values)
        if(.not. status) call lib$signal(%val(unexperr), %val(1),
     &    'OVERLAY', %val(status))
        axes_flags = flags .and. axes_mask
*
*       Get the  axes ranges,  and, if  manual scaling specifed,
```

```
*       offer the user the scaling choice.
*
        xmin = x(1,1)
        xmax = x(1,1)
        ymin = y(1,1)
        ymax = y(1,1)
        do n = 1, ncurv
          do m = 1, npts(n)
            if(x(m,n) .lt. xmin) xmin = x(m,n)
            if(x(m,n) .gt. xmax) xmax = x(m,n)
            if(y(m,n) .lt. ymin) ymin = y(m,n)
            if(y(m,n) .gt. ymax) ymax = y(m,n)
          end do
        end do

        if(manualx) then                          ! manual scaling X
          sxmin = dsplay$xmin
          sxmax = dsplay$xmax
          incx = dsplay$incx
        else
          call scal(xmin, xmax, sxmin, sxmax, incx)
        endif

        if(manualy) then                          ! manual scaling Y
          symin = dsplay$ymin
          symax = dsplay$ymax
          incy = dsplay$incy
        else
          call scal(ymin, ymax, symin, symax, incy)
        endif
        if(incx .eq. 0 .or. incy .eq. 0) goto 50

        if(.not. laser) then
          call quiet_plot('on')                   ! set terminal /NOBROADCAST
          if(vt240 .and. autohc) then
            if(rotate) then
              status = lib$put_screen(esc // '[?47h')
            else
              status = lib$put_screen(esc // '[?47l')
            endif
            if(.not. status) call lib$signal(%val(plt$_unexperr), %val(1),
     &        'OVERLAY', %val(status))
          endif
          if(vt240) call vt200_set_mode(4)        ! TEK emulation
        endif
        if(first) then
          call initt(baudq()/10)                  ! In chars/sec
          first = .false.
        elseif(laser) then
          call ff_laser_plot
        else
          call newpag                             ! clear screen
        endif

        call dsp_wpage(16., 14., 8., 8.)          ! Physical window
```

```
      call axes1(sxmin,sxmax,incx,symin,symax,incy,axes_flags)
      lttl = llen(ttl)
      lx = llen(xlabl)
      ly = llen(ylabl)
      call mtitle(ttl, lttl, xlabl, lx, ylabl, ly)
      jsym = 0
      call n_curve(x, y, npts, ncurv, maxpts, jsym)
      if(.not. laser) then
        if(autohc) then
          call hard_copy                        ! Hardcopy
        endif

        if(vt240) then
          call vt200_set_mode(5)                ! Back to VT240
          if(autohc) call hard_copy_ff          ! ff printer if necessary
          call recolor                          ! change vt240 color back
        endif
        if(rotate) then
          status = lib$put_screen(esc // '[?47l')
          if(.not. status) call lib$signal(%val(plt$_unexperr), %val(1),
     &      'OVERLAY', %val(status))
        endif
      endif
      if(.not. laser) call quiet_plot('off')

50    return
      end
```

SUBROUTINE NCKNEW (NECK)

```
C       NCKNEW.FOR
C       7/12/83. J.B. MATRIX 'F' IS CORRECTED/12 CHANGES
C
C
C       VRBLS()=LIST OF VARIABLES' NAMES
C       FILNM=FILE NAME
C       NAMTMP()=RUN NUMBER
C       NP()=NUMBER OF POINTS OF OBSERVATIONS
C       NERROR=AN ERROR INDICATOR
C       RECNM()=THE RECORD NUBER IN SCRATCH FILE SCRTCH.DAT
C       TYPE()=ERROR TYPE
C       RECLN=LENGTH OF A RECORD (NUMBER OF WORDS)
C       DATA RECLN/801/=RECORD LENGTH
C       NVAR=NUMBER OF VARIABLES
C       NAME(NVAR)=THE VARIABLE'S NAME
C       RUN(NVAR)=RUN NUMBER
C       MAX(NVAR)=THE MAXIMUM OF THAT VARIABLE
C       MIN(NVAR)=THE MINIMUM OF THAT VARIABLE
C       UNITS(NVAR)=A UNIT OF MEASUREMENT
C       NUMB(NVAR)=NUMBER OF POINTS OF OBSERVATIOMS
C       NUM=NP(MAX INDEX)=NUMBER OF POINTS OF OBSERVATIONS
C
        BYTE UNITS(100)
        INTEGER*2 NUMB(100)
        INTEGER RECLEN,MREC,NP(13),NERROR,RECNM(13),TYPE(2)
        REAL DAXSOP(600),DAYSOP(600),DAZSOP(600),DNXSOP(600),
     &    DNYSOP(600),DNZSOP(600),PHAOXP(600),PHBO2P(600),PHCO3P(600),
     &    PNAOXP(600),PNBO2P(600),PNCO3P(600),TARRY(600),MAX(100),
     &    MIN(100)
        CHARACTER*6 NAME(100),RUN(100),VRBLS(13),FILNM,NAMTMP

        COMMON /INDATA/ DAXSOP,DAYSOP,DAZSOP,DNXSOP,DNYSOP,DNZSOP,
     &    PHAOXP,PHBO2P,PHCO3P,PNAOXP,PNBO2P,PNCO3P,TARRY

        PARAMETER (MREC=101)
        PARAMETER (RECLEN=598)

         DATA VRBLS/'DAXSOP','DAYSOP','DAZSOP','DNXSOP','DNYSOP',
     &    'DNZSOP','PHAOXP','PHBO2P','PHCO3P','PNAOXP','PNBO2P',
     &    'PNCO3P','TIME'/
C
        DATA TYPE/'OPEN','READ'/
C
C
C  STEP 1: OPEN INPUT FILE
C
C
        OPEN(UNIT=1,FILE='SCRTCH',STATUS='OLD',RECL=RECLEN,ERR=999,
     &    FORM='UNFORMATTED',ORGANIZATION='RELATIVE',IOSTAT=IOS,
     &    ACCESS='DIRECT')
C
C
C  STEP 2: READ IN DIRECTORY
C
C
```

```
      READ(1,REC=1)NVAR,NAME,RUN,MAX,MIN,UNITS,NUMB
C
C
C
C  STEP 3: TEST FOR EXISTANCE OF VARIABLES
C
C
      DO 1 I=1,13
      DO 2 N=1,NVAR
      IF(NAME(N).NE.VRBLS(I))GO TO 3
      NP(I)=NUMB(N)
      RECNM(I)=N+1
      GO TO 1
 3        IF(N.EQ.NVAR)GO TO 900
 2     CONTINUE
 1    CONTINUE
C
C
C  STEP 4: READ APPROPRIATE RECORD
C
C
      READ(1,REC=RECNM(1))(DAXSOP(K),K=1,NP(1))
      READ(1,REC=RECNM(2))(DAYSOP(K),K=1,NP(2))
      READ(1,REC=RECNM(3))(DAZSOP(K),K=1,NP(3))
      READ(1,REC=RECNM(4))(DNXSOP(K),K=1,NP(4))
      READ(1,REC=RECNM(5))(DNYSOP(K),K=1,NP(5))
      READ(1,REC=RECNM(6))(DNZSOP(K),K=1,NP(6))
      READ(1,REC=RECNM(7))(PHAOXP(K),K=1,NP(7))
      READ(1,REC=RECNM(8))(PHBO2P(K),K=1,NP(8))
      READ(1,REC=RECNM(9))(PHCO3P(K),K=1,NP(9))
      READ(1,REC=RECNM(10))(PNAOXP(K),K=1,NP(10))
      READ(1,REC=RECNM(11))(PNBO2P(K),K=1,NP(11))
      READ(1,REC=RECNM(12))(PNCO3P(K),K=1,NP(12))
      READ(1,REC=RECNM(13))(TARRY(K),K=1,NP(13))
C
 10   CONTINUE
      NERROR=0
      NUM=NP(13)
C
C
      NAMTMP=RUN(RECNM(1)-1)
      CALL NECKTP(NUM,NERROR,NAMTMP)
C
      IF(NERROR.NE.0)GO TO 999
      GO TO 1000
C
C
C
C
C
C
C
C  ERROR MESSAGES
 900   WRITE(6,901)VRBLS(I)
 901   FORMAT(1X,'VARIABLE',1X,A10,1X,'IS NOT IN THE INPUT FILE',/,
```

```
     +         1X,'YOU ARE ABOUT TO BE RETURNED TO MONITOR LEVEL TO ',/,
     +         1X,'TRY AND RECTIFY THE PROBLEM',///)
C
C
C
         GO TO 1000
C
C
C
 999   WRITE(6,902)TYPE(NERROR)
C
 902   FORMAT(1X,A5,'ERROR ENCOUNTERED'/,
     +      1X,'PROGRAM HALT',///)
C
 1000  CONTINUE
C
         CLOSE(UNIT=1)
         END
*******************************************************************************
*******************************************************************************
         SUBROUTINE NECKTP(NUM,NERROR,RUNTMP)
C        4TH EDITION OF 2/22/83
C
C        FILE:  NECKTP4.FOR
C
C        COMPUTATION OF NECK STRETCH T1 TO OCCIPITAL CONDYLES
C                    FROM PHOTOGRAPHIC DATA
C
C        GLOSSARY
C
C        PROGRAM CONSTANTS
C
C        RGAX = THE COMPONENT OF LINEAR POSITION OF THE
C                HEAD CENTER OF GRAVITY ALONG THE X-AXIS
C                OF THE HEAD ANATOMICAL COORDINATE SYSTEM.
C        RGAZ = SAME AS ABOVE EXCEPT FOR THE Z-AXIS.
C        RGOX = THE COMPONENT OF LINEAR POSITION OF THE
C                HEAD CENTER OF GRAVITY ALONG THE X-AXIS OF THE
C                HEAD ANATOMICAL COORD. SYSTEM MEASURED FROM
C                THE OUTSIDE CONDYLES.
C        RGOZ = SAME AS ABOVE EXCEPT FOR THE Z-AXIS.
C
C        PROGRAM VARIABLES (ARRAYS)
C
C        DAXSOP = X-COMPONENT OF DISPLACEMENT OF HEAD A.O.
C                  (THE SLED COORDINATE SYETEM) FROM
C                  PHOTOGRAPHIC DATA
C        DAYSOP = SAME AS ABOVE EXCEPT FOR THE Y-COMPONENT
C        DAZSOP = SAME AS ABOVE EXCEPT FOR THE Z-COMPONENT
C        DNXSOP = X-COMPONENT OF DISPLACEMENT OF THE T1
C                  A.O. (THE SLED COORDINATE SYSTEM) FROM
C                  PHOTOGRAPHIC DATA
C        DNYSOP = SAME AS ABOVE EXCEPT FOR THE Y-COMPONENT
C        DNZSOP = SAME AS ABOVE EXCEPT FOR THE Z-COMPONENT
C        PHAOXP = HEAD ROTATION ABOUT X AXIS (THE ANATOMIC
```

```
C               COORD. SYSTEM) FROM PHOTOGRAPHIC DATA
C       PHB02P = SAME AS ABOVE EXCEPT FOR THE Y-COMPONENT
C       PHC03P = SAME AS ABOVE EXCEPT FOR THE Z-COMPONENT
C       PNA0XP = ANGLE OF ROTATION OF T1 ABOUT X AXIS OF THE T1
C               ANATOMICAL COORD. SYSTEM AS DERIVED FROM
C               PHOTOGRAPHIC DATA
C       PNB02P = SAME AS ABOVE EXCEPT ABOUT THE CARRIED Y AXIS
C       PNC03P = SAME AS ABOVE EXCEPT ABOUT THE CARRIED Z AXIS
C       TOXLP/TOYLP/TOZLP=OUTPUT VARIABLES GENERATED BY
C                               PROGRAM TRQPHO.FOR :
C                       THE COMPONENT OF THE MOMENT APPLIED BY THE
C                       NECK TO THE HEAD ABOUT AN AXIS PARALLEL
C                       TO THE LABORATORY X/Y/Z-AXIS AND PASSING
C                       THROUGH THE ORIGIN OF THE OCCIPITAL COORD.
C                       SYSTEM.
C       FOXLP/FOYLP/FOZLP= OUTPUT VARIABLES FROM PROGRAM
C                       TRQPHO.FOR:
C                       THE COMPONENT OF FORCE APPLIED BY THE NECK
C                       TO THE HEAD PARALLEL TO THE LABORATORY
C                       X/Y/Z-AXIS AND PASSING THROUGH THE ORIGIN
C                       OF THE OCCIPITAL COORD. SYSTEM.
C
C       OUTPUT VARIABLES
C
C       RATIXP = X-COMPONENT OF POSITION OF THE HEAD ANATOMICAL
C               ORIGIN WITH RESPECT TO THE T1 ANATOMICAL ORIGIN
C               THE LABORATORY COORD. SYSTEM) FROM PHOTOGRAPHIC
C               DATA
C       RATIYP = SAME AS ABOVE EXCEPT FOR THE Y-COMPONENT
C       RATIZP = SAME AS ABOVE EXCEPT FOR THE Z-COMPONENT
C       ROTIXP = X-COMPONENT OF POSITION OF THE OCCIPITAL
C               CONDYLE WITH RESPECT TO THE T1 ANATOMIC
C               ORIGIN (THE LABORATORY COORD. SYSTEM) FROM
C               PHOTOGRAPHIC DATA
C       ROTIYP = SAME AS ABOVE EXCEPT FOR THE Y-COMPONENT
C       ROTIZP = SAME AS ABOVE EXCEPT FOR THE Z-COMPONENT
C       RATIP = THE DISTANCE FROM THE HEAD ANATOMICAL
C               ORIGIN TO THE T1 ANATOMICAL ORIGIN
C               FROM PHOTOGRAPHIC DATA
C       ROTIP = THE DISTANCE FROM THE OCCIPITAL
C               CONDYLE TO THE T1 ANATOMICAL ORIGIN
C               FROM PHOTOGRAPHIC DATA
C       TENTYP = THE ANGLE OF ROTATION OF A PLANE FORMED BY THE
C               Y AXIS OF THE T1 ANATOMICAL COORD. SYSTEM AND
C               THE VECTOR JOINING T1 WITH THE
C               OCCIPITAL CONDYLE WITH RESPECT TO THE PLANE FORMED
C               BY THE Y AND Z AXES OF THE T1 ANATOMICAL COORD.
C               SYSTEM AS DERIVED FROM PHOTOGRAPHIC DATA
C       TENTXP = THE ANGLE OF ROTATION OF A PLANE FORMED BY THE
C               X AXIS OF THE T1 ANATOMICAL COORD. SYSTEM AND THE
C               VECTOR JOINING T1 WITH RESPECT TO THE PLANE FORMED
C               BY THE X AND Z AXES OF THE T1 ANATOMICAL COORD.
C               SYSTEM AS DERIVED FROM PHOTOGRAPHIC DATA
C       TETHNC=THE ANGLE OF ROTATION OF A PLANE FORMED BY THE
C              X-AXIS OF THE HEAD ANATOMICAL ORIGIN AND
```

```
C                THE Z-AXIS OF THE NECK CHORD LINE COORDINATE SYSTEM
C                WITH RESPECT TO THE PLANE FORMED BY THE X-
C                AND Z-AXES OF THE NECK LINE COORDINATE SYSTEM
C        TOXOP = THE COMPONENT OF TORQUE APPLIED BY THE NECK TO THE
C                HEAD AT THE OCCIPITAL CONDYLES ALONG THE X-AXIS
C                OF THE NECK CHORD COORDINATE SYSTEM
C        TOYOP = THE COMPONENT OF TORQUE APPLIED BY THE NECK
C                TO THE HEAD AT THE OCCIPITAL CONDYLES ALONG THE-
C                AXIS OF THE NECK CHORTD COORDINATE SYSTEM
C        TOZOP = THE COMPONENT OF TORQUE APPLIED BY THE NECK TO
C                THE HEAD AT THE OCCIPITAL CONDYLES ALONG THE Z-AXIS
C                OF THE NECK CHORD COORDINATE SYSTEM
C        FOXOP = THE COMPONENT OF FORCE APPLIED BY THE NECK TO THE
C                HEAD AT THE OCCIPITAL CONDYLES ALONG THE X-AXIS
C                OF THE NECK CHORD COORDINATE SYSTEM
C        FOYOP = THE COMPONENT OF FORCE APPLIED BY THE NECK TO THE
C                HEAD AT THE OCCIPITAL CONDYLES ALONG THE Y-AXIS
C                OF THE NECK CHORD COORDINATE SYSTEM
C        FOZOP = THE COMPONENT OF FORCE APPLIED BY THE NECK TO THE
C                HEAD AT THE OCCIPITAL CONDYLES ALONG THE Z-AXIS
C                OF THE NECK CHORD COORD. SYSTEM
C        T1XLP= THE COMPONENT OF TORQUE APPLIED BY THE TORSO TO THE NECK
C                AT THE T1 VERTEBRA ALONG THE X-AXIS OF THE LABORATORY
C                COORDINATE SYSTEM.
C        T1YLP= THE COMPONENT OF TORQUE APPLIED BY THE TORSO TO THE NECK
C                AT THE T1 VERTEBRA ALONG THE Y-AXIS OF THE  LABORATORY
C                COORDINATE SYSTEM.
C        T1ZLP= THE COMPONENT OF TORQUE APPLIED BY THE TORSO TO THE NECK
C                AT THE T1 VERTEBRA ALONG THE Z-AXIS OF THE LABORATAORY
C                COORDINATE SYSTEM.
C
         REAL TOXLP(600),TOYLP(600),TOZLP(600),FOXLP(600),FOYLP(600),
     &     FOZLP(600),TOXOP(600),TOYOP(600),TOZOP(600),FOXOP(600),
     &     FOYOP(600),FOZOP(600),TETHNP(600),PSI(600),RATIXP(600),
     &     RATIYP(600),RATIZP(600),ROTIXP(600),ROTIYP(600),ROTIZP(600),
     &     RATIP(600),ROTIP(600),TENTYP(600),TENTXP(600),T1XLP(600),
     &     T1YLP(600),T1ZLP(600),T1XTP(600),T1YTP(600),T1ZTP(600)
         CHARACTER*6 NAMTMP(24),RUNTMP
         INTEGER NERROR
         INCLUDE 'COMP.FOR'
C
C        TRANSFER OF DATA FROM TRQPHO.FOR PROGRAM
C
C
         DATA NAMTMP/'RATIXP','RATIYP','RATIZP','ROTIXP','ROTIYP',
     &     'ROTIZP','RATIP','ROTIP','TENTYP','TENTXP','TETHNP',
     &     'PSI','TOXOP','TOYOP','TOZOP','FOXOP','FOYOP','FOZOP',
     &     'T1XLP','T1YLP','T1ZLP','T1XTP','T1YTP','T1ZTP'/
C        SUBJECT NUMBER SELECTION
1700     CONTINUE
         READ(5,*)NJCT
         WRITE(6,1755)NJCT
1755     FORMAT(5X,'SUBJECT NUMBER=',I5)
         IF(NJCT.EQ.1)GO TO 1001
         IF(NJCT.EQ.44)GOTO 1044
```

```
      IF(NJCT.EQ.64)GOTO 1064
      IF(NJCT.EQ.65)GOTO 1065
      IF(NJCT.EQ.67)GOTO 1067
      IF(NJCT.EQ.83)GO TO 1083
      IF(NJCT.EQ.93)GO TO 1093
      IF(NJCT.EQ.96)GO TO 1096
      IF(NJCT.EQ.118)GO TO 1118
      IF(NJCT.EQ.120)GO TO 1120
      IF(NJCT.EQ.127)GO TO 1127
      IF(NJCT.EQ.130)GO TO 1130
      IF(NJCT.EQ.131)GO TO 1131
      IF(NJCT.EQ.132)GO TO 1132
      IF(NJCT.EQ.133)GO TO 1133
      IF(NJCT.EQ.134)GO TO 1134
      IF(NJCT.EQ.135)GO TO 1135
      IF(NJCT.EQ.136)GO TO 1136
      IF(NJCT.EQ.138)GO TO 1138
      IF(NJCT.EQ.139)GO TO 1139
      IF(NJCT.EQ.140)GO TO 1140
      IF(NJCT.EQ.141)GO TO 1141
      IF(NJCT.EQ.142)GO TO 1142
1701  CONTINUE
      WRITE(6,1702)
1702  FORMAT(1X,'INCORRECT SUBJECT NUMBER')
      STOP
1001  CONTINUE
      RGAX=0.012
      RGAZ=0.029
      RGOX=0.0234
      RGOZ=0.055
      ARP=0.0
      BR=0.0
      DNZMN=0.0
      XCR=0.0
      YCR=0.
      ZCR=0.0
      GO TO 999
C
1044  CONTINUE
C
      RGAX=0.012
      RGAZ=0.029
      RGOX=0.023
      RGOZ=0.055
      ARP=1.160
      BR=-0.492
      DNZMN=1.091
      XCR=0.0
      YCR=0.0
      ZCR=0.0
      GO TO 999
C
1064  CONTINUE
C
      RGAX=0.012
```

```
      RGAZ=0.029
      RGOX=0.023
      RGOZ=0.055
      ARP=1.146
      BR=-0.294
      DNZMN=1.091
      XCR=0.
      YCR=0.0
      ZCR=0.0
      GO TO 999
1065  CONTINUE
C
      RGAX=0.012
      RGAZ=0.029
      RGOX=0.023
      RGOZ=0.055
      ARP=1.154
      BR=-0.410
      DNZMN=1.095
      XCR=0.
      YCR=0.0
      ZCR=0.0
      GO TO 999
1067  CONTINUE
C
      RGAX=0.012
      RGAZ=0.029
      RGOX=0.023
      RGOZ=0.055
      ARP=1.059
      BR=-0.0719
      DNZMN=1.047
      XCR=0.0
      YCR=0.0
      ZCR=0.0
      GO TO 999
C
1083  CONTINUE
C
      RGAX=0.012
      RGAZ=0.029
      RGOX=0.023
      RGOZ=0.055
      ARP=1.272224
      BR=-1.31925
      DNZMN=1.128304
      XCR=0.0
      YCR=0.0
      ZCR=0.0
      GO TO 999
C
1093  CONTINUE
      RGAX=0.012
      RGAZ=0.029
      RGOX=0.023
```

```
      RGOZ=0.055
      ARP=1.239763
      BR=-0.941244
      DNZMN=1.098385
      XCR=0.0
      YCR=0.
      ZCR=0.
      GO TO 999
C
1096  CONTINUE
      RGAX=0.012
      RGAZ=0.029
      RGOX=0.023
      RGOZ=0.055
      GO TO 999
 1118 CONTINUE
      RGAX=.012
      RGAZ=.029
      RGOX=.023
      RGOZ=.055
      ARP=1.472471
      BR=-.537460
      DNZMN=1.380167
      XCR=0.
      YCR=0.
      ZCR=0.
      GO TO 999
 1120 CONTINUE
      RGAX=.012
      RGAZ=.029
      RGOX=.023
      RGOZ=.055
      ARP=1.554568
      BR=-1.00074
      DNZMN=1.383071
      XCR=0.
      YCR=0.
      ZCR=0.
      GO TO 999
 1127 CONTINUE
      RGAX=.012
      RGAZ=.029
      RGOX=.023
      RGOZ=.055
      ARP=1.541470
      BR=-.976451
      DNZMN=1.382870
      XCR=0.
      YCR=0.
      ZCR=0.
      GO TO 999
 1130 CONTINUE
      RGAX=.012
      RGAZ=.029
      RGOX=.023
```

```
      RGOZ=.055
      ARP=1.665381
      BR=-1.48895
      DNZMN=1.396958
      XCR=0.
      YCR=0.
      ZCR=0.
      GO TO 999
1131  CONTINUE
      RGAX=.012
      RGAZ=.029
      RGOX=.023
      RGOZ=.055
      ARP=1.520460
      BR=-0.753537
      DNZMN=1.402846
      XCR=0.
      YCR=0.
      ZCR=0.
      GO TO 999
1132  CONTINUE
      RGAX=.012
      RGAZ=0.029
      RGOX=.023
      RGOZ=.055
      ARP=1.523568
      BR=-0.927588
      DNZMN=1.392558
      XCR=0.
      YCR=0.
      ZCR=0.
      GO TO 999
1133  CONTINUE
      RGAX=.012
      RGAZ=.029
      RGOX=.023
      RGOZ=.055
      ARP=1.513768
      BR=-0.751378
      DNZMN=1.389440
      XCR=0.
      YCR=0.
      ZCR=0.
      GO TO 999
1134  CONTINUE
      RGAX=.012
      RGAZ=.029
      RGOX=.023
      RGOZ=.055
      ARP=1.543767
      BR=-0.862095
      DNZMN=1.40791
      XCR=0.
      YCR=0.
      ZCR=0.
```

```
      GO TO 999
1135  CONTINUE
      RGAX=.012
      RGAZ=.029
      RGOX=.023
      RGOZ=.055
      ARP=1.552383
      BR=-0.962941
      DNZMN=1.408047
      XCR=0.
      YCR=0.
      ZCR=0.
      GO TO 999
1136  CONTINUE
      RGAX=.012
      RGAZ=.029
      RGOX=.023
      RGOZ=.055
      ARP=1.413834
      BR=-0.130286
      DNZMN=1.391344
      XCR=0.
      YCR=0.
      ZCR=0.
      GO TO 999
1138  CONTINUE
      RGAX=.012
      RGAZ=.029
      RGOX=.023
      RGOZ=.055
      ARP=1.493554
      BR=-0.626753
      DNZMN=1.384522
      XCR=0.
      YCR=0.
      ZCR=0.
      GO TO 999
1139  CONTINUE
      RGAX=.012
      RGAZ=.029
      RGOX=.023
      RGOZ=.055
      ARP=1.537098
      BR=-0.821211
      DNZMN=1.401846
      XCR=0.
      YCR=0.
      ZCR=0.
      GO TO 999
1140  CONTINUE
      RGAX=.012
      RGAZ=.029
      RGOX=.023
      RGOZ=.055
      ARP=1.515952
```

```
      BR=-0.732621
      DNZMN=1.388380
      XCR=0.
      YCR=0.
      ZCR=0.
      GO TO 999
1141  CONTINUE
      RGAX=0.012
      RGAZ=.029
      RGOX=.023
      RGOZ=.055
      ARP=1.542182
      BR=-0.785473
      DNZMN=1.403853
      XCR=0.
      YCR=0.
      ZCR=0.
      GO TO 999
1142  CONTINUE
      RGAX=.012
      RGAZ=.029
      RGOX=.023
      RGOZ=.055
      ARP=1.558605
      BR=-0.987192
      DNZMN=1.399655
      XCR=0.
      YCR=0.
      ZCR=0.
      GO TO 999
C
1725  CONTINUE
999   CONTINUE
      DELT=0.0005
!     TPMAX=TARRY(NUM)
      TSMAX=DELT*598
!     IF(TPMAX.LE.TSMAX)GO TO 55
!     print *,'here I am'
      DO 11 K=1,NUM
      K1=K
      TAR=TARRY(K1)
      IF(TAR.LE.TSMAX)GO TO 11
      KC=K
      KMAX=KC-1
      GO TO 22
C
11    CONTINUE
22    CONTINUE
C
!     print *,'kmax=',kmax
      NUM=KMAX
C     OPEN INPUT FILENNN.DAT
C
      OPEN(UNIT=10,FILE='NNN',ACCESS='SEQUENTIAL',STATUS='OLD')
      DO 1212 K=1,NUM
```

```
      J=K
      READ(10,1220)TOXLP(J),TOYLP(J),TOZLP(J),
     +                  FOXLP(J),FOYLP(J),FOZLP(J)
C     write(6,1220)toxlp(j)
1220  FORMAT(6(E13.7))
1212  CONTINUE
      CLOSE(UNIT=10)
C
55    CONTINUE
C     EQUATIONS (2)
C     ***************

      ROAXA=RGAX-RGOX
      ROAZA=RGAZ-RGOZ
C
C     MATRIX P
C
      TET1XO=PNAOXP(1)
      TET1YO=PNBO2P(1)
      TET1ZO=PNCO3P(1)
C
      P11=COS(TET1ZO)*COS(TET1YO)
            P211=COS(TET1XO)*SIN(TET1ZO)
            P212=COS(TET1ZO)*SIN(TET1YO)*SIN(TET1XO)
      P21=P211+P212
            P311=SIN(TET1ZO)*SIN(TET1XO)
            P312=-COS(TET1ZO)*SIN(TET1YO)*COS(TET1XO)
      P31=P311+P312
C
      P12=-SIN(TET1ZO)*COS(TET1YO)
            P221=COS(TET1ZO)*COS(TET1XO)
            P222=-SIN(TET1ZO)*SIN(TET1YO)*SIN(TET1XO)
      P22=P221+P222
            P321=COS(TET1ZO)*SIN(TET1XO)
            P322=SIN(TET1ZO)*SIN(TET1YO)*COS(TET1XO)
      P32=P321+P322
C
      P13=SIN(TET1YO)
      P23=-COS(TET1YO)*SIN(TET1XO)
      P33=COS(TET1YO)*COS(TET1XO)
C
      I=0
10    CONTINUE
      I=I+1
      IF(I.EQ.NUM)GO TO 100
      TETAX=PHAOXP(I)
      TETAY=PHBO2P(I)
      TETAZ=PHCO3P(I)
C
C     EQUATIONS (3) -- TRANSFORMATION OF OCCIPITAL
C     CONDYLE LOCATION RELATIVE TO ANATOMICAL ORIGIN IN
C     THE ANATOMICAL COORDINATE SYSTEM TO COMPONENTS IN
C     IN THE LABORATORY COORD. SYST.
C     ******************************
            A11=ROAXA*COS(TETAZ)*COS(TETAY)
```

```
            A12=ROAZA*SIN(TETAY)
        ROAXP=A11+A12
            A21=ROAXA*COS(TETAX)*SIN(TETAZ)
            A22=ROAXA*COS(TETAZ)*SIN(TETAY)*SIN(TETAX)
            A23=-ROAZA*COS(TETAY)*SIN(TETAX)
        ROAYP=A21+A22+A23
            A31=ROAXA*SIN(TETAZ)*SIN(TETAX)
            A32=-ROAXA*COS(TETAZ)*SIN(TETAY)*COS(TETAX)
            A33=ROAZA*COS(TETAY)*COS(TETAX)
        ROAZP=A31+A32+A33
C
            RAXP=DAXSOP(I)
            RAYP=DAYSOP(I)
            RAZP=DAZSOP(I)
            RTIXP=DNXSOP(I)
            RTIYP=DNYSOP(I)
        RTIZP=DNZSOP(I)
C
C       EQUATIONS (6)
C       **************
C
        B1=RAXP-RTIXP
        B2=RAYP-RTIYP
        B3=RAZP-RTIZP
        RATIXP(I)=B1
        RATIYP(I)=B2
        RATIZP(I)=B3
        RATIP(I)=SQRT(B1**2+B2**2+B3**2)
        IF(I.GT.1)GO TO 200
        YP=ARP+BR*RATIP(1)
        DL2=YP-DNZMN
200     CONTINUE
        B1P=B1+XCR
        B2P=B2+YCR
        B3P=B3+DL2+ZCR
        RATIP(I)=SQRT(B1P**2+B2P**2+B3P**2)
C
C       EQUATIONS (7)
C
        C1=B1P+ROAXP
        C2=B2P+ROAYP
        C3=B3P+ROAZP
        ROTIXP(I)=C1
        ROTIYP(I)=C2
        ROTIZP(I)=C3
        ROTIP(I)=SQRT(C1**2+C2**2+C3**2)
1899    CONTINUE ! TEMPORARY TO 1901
        IF(ROTIP(I).LT.0.25)GO TO 1901
        RATIP(I)=0.
!       ROTIP(I)=0.
        WRITE(6,1900)
1900    FORMAT(1X,'MISSING PHOTO DATA?')
1901    CONTINUE
C       EQUATIONS (9)
C       **************
```

```
C
      TETA1X=PNA0XP(I)
      TETA1Y=PNB02P(I)
      TETA1Z=PNC03P(I)
      PSI(I)=1.5708-PHC03P(I)
C
C     MATRIX D
C
      D11=COS(TETA1Z)*COS(TETA1Y)
      D21=-SIN(TETA1Z)*COS(TETA1Y)
      D31=SIN(TETA1Y)
            D121=COS(TETA1X)*SIN(TETA1Z)
            D122=COS(TETA1Z)*SIN(TETA1Y)*SIN(TETA1X)
      D12=D121+D122
            D221=COS(TETA1Z)*COS(TETA1X)
            D222=-SIN(TETA1Z)*SIN(TETA1Y)*SIN(TETA1X)
      D22=D221+D222
      D32=-COS(TETA1Y)*SIN(TETA1X)
            D131=SIN(TETA1Z)*SIN(TETA1X)
            D132=-COS(TETA1Z)*SIN(TETA1Y)*COS(TETA1X)
      D13=D131+D132
            D231=COS(TETA1Z)*SIN(TETA1X)
            D232=SIN(TETA1Z)*SIN(TETA1Y)*COS(TETA1X)
      D23=D231+D232
      D33=COS(TETA1Y)*COS(TETA1X)
C     EQUATIONS (10)
C     **************
C     MATRIX PD=P X D
C
      PD11=P11*D11+P12*D21+P13*D31
      PD21=P21*D11+P22*D21+P23*D31
      PD31=P31*D11+P32*D21+P33*D31
C
      PD12=P11*D12+P12*D22+P13*D32
      PD22=P21*D12+P22*D22+P23*D32
      PD32=P31*D12+P32*D22+P33*D32
C
      PD13=P11*D13+P12*D23+P13*D33
      PD23=P21*D13+P22*D23+P23*D33
      PD33=P31*D13+P32*D23+P33*D33
C
      ROTX=PD11*C1+PD12*C2+PD13*C3
      ROTY=PD21*C1+PD22*C2+PD23*C3
      ROTZ=PD31*C1+PD32*C2+PD33*C3
C
      GAMA=ASIN(ROTY/ROTIF(I))
      TENTX=GAMA

C     IF ROTZ IS APPROACHING ZERO THEN ATAN IS 90 DEGREES
      IF(ABS(ROTZ).LT.0.000001)GO TO 1000
      BETA=ATAN(ROTX/ROTZ)
      IF((ROTX.GE.0).AND.(ROTZ.GT.0))TENTY=BETA
      IF((ROTX.GE.0).AND.(ROTZ.LT.0))TENTY=3.14158+BETA
      IF((ROTX.LT.0).AND.(ROTZ.GT.0))TENTY=BETA
      IF((ROTX.LT.0).AND.(ROTZ.LT.0))TENTY=-(3.14158-BETA)
```

```
C       GAMA=ATAN(ROTY/ROTIP(I))
C       IF((ROTY.GE.0).AND.(ROTZ.GT.0))TENTX=GAMA
C       IF((ROTY.GE.0).AND.(ROTZ.LT.0))TENTX=3.14158+GAMA
C       IF((ROTY.LT.0).AND.(ROTZ.GT.0))TENTX=GAMA
C       IF((ROTY.LT.0).AND.(ROTZ.LT.0))TENTX=-(3.14158-GAMA)
        GO TO 2000
1000    CONTINUE
        IF(ROTX.GE.0)SIGN1=1
        IF(ROTX.LT.0)SIGN1=-1
        IF(ROTY.GE.0)SIGN2=1
        IF(ROTY.LT.0)SIGN2=-1
        TENTY=SIGN1*1.57
C       TENTX=SIGN2*1.57
2000    CONTINUE
        TENTYP(I)=TENTY
        IF(I.GT.1) GO TO 2001
        TENTY1=TENTY
 2001   TENTYP(I) = TENTYP(I) - TENTY1
        TENTXP(I)=TENTX
C
C       EQUATION (14)
C       **************
C
        TET11Y=ASIN(ROTIXP(I)/ROTIP(I))
C
C       EQUATION (15)
C       **************
C
        TET11X=-TENTX
C
C       EQUATIONS (19)
C       ***************
C
C       MATRIX RO
C
        RO11=COS(TET11Y)
        RO21=0.
        RO31=SIN(TET11Y)
C
        RO12=SIN(TET11Y)*SIN(TET11X)
        RO22=COS(TET11X)
        RO32=-COS(TET11Y)*SIN(TET11X)
C
        RO13=-SIN(TET11Y)*COS(TET11X)
        RO23=SIN(TET11X)
        RO33=COS(TET11Y)*COS(TET11X)
C
C       MATRIX ROD = RO X D
C
        ROD11=RO11*D11+RO12*D21+RO13*D31
        ROD21=RO21*D11+RO22*D21+RO23*D31
        ROD31=RO31*D11+RO32*D21+RO33*D31
C
        ROD12=RO11*D12+RO12*D22+RO13*D32
        ROD22=RO21*D12+RO22*D22+RO23*D32
```

```
      ROD32=RO31*D12+RO32*D22+RO33*D32
C
      ROD13=RO11*D13+RO12*D23+RO13*D33
      ROD23=RO21*D13+RO22*D23+RO23*D33
      ROD33=RO31*D13+RO32*D23+RO33*D33
C
      TOXO=ROD11*TOXLP(I)+ROD12*TOYLP(I)+ROD13*TOZLP(I)
      TOYO=ROD21*TOXLP(I)+ROD22*TOYLP(I)+ROD23*TOZLP(I)
      TOZO=ROD31*TOXLP(I)+ROD32*TOYLP(I)+ROD33*TOZLP(I)
C
      TOXOP(I)=TOXO
      TOYOP(I)=TOYO
      TOZOP(I)=TOZO
C
C     EQUATIONS (20)
C     ******************
C
      FOXO=ROD11*FOXLP(I)+ROD12*FOYLP(I)+ROD13*FOZLP(I)
      FOYO=ROD21*FOXLP(I)+ROD22*FOYLP(I)+ROD23*FOZLP(I)
      FOZO=ROD31*FOXLP(I)+ROD32*FOYLP(I)+ROD33*FOZLP(I)
C
      FOXOP(I)=FOXO
      FOYOP(I)=FOYO
      FOZOP(I)=FOZO
      T1XLP(I)= TOXLP(I) + ROTIYP(I)*FOZLP(I) - ROTIZP(I)*FOYLP(I)
      T1YLP(I)= TOYLP(I) - ROTIXP(I)*FOZLP(I) + ROTIZP(I)*FOXLP(I)
      T1ZLP(I)= TOZLP(I) + ROTIXP(I)*FOYLP(I) - ROTIYP(I)*FOXLP(I)
      T1XTP(I)= PD11*T1XLP(I)+PD12*T1YLP(I)+PD13*T1ZLP(I)
      T1YTP(I)= PD21*T1XLP(I)+PD22*T1YLP(I)+PD23*T1ZLP(I)
      T1ZTP(I)= PD31*T1XLP(I)+PD32*T1YLP(I)+PD33*T1ZLP(I)
C
C     EQUATIONS (17)
C     **************
C     MATRIX R
C
      R11=COS(TET11Y)
      R21=0.
      R31=SIN(TET11Y)
C
      R12=SIN(TET11Y)*SIN(TET11X)
      R22=COS(TET11X)
      R32=-COS(TET11Y)*SIN(TET11X)
C
      R13=-SIN(TET11Y)*COS(TET11X)
      R23=SIN(TET11X)
      R33=COS(TET11Y)*COS(TET11X)
C
C     MATRIX F
C
      F1=COS(TETAZ)*COS(TETAY)
              F21=COS(TETAX)*SIN(TETAZ)
              F22=COS(TETAZ)*SIN(TETAY)*SIN(TETAX)
      F2=F21+F22
              F31=SIN(TETAZ)*SIN(TETAX)
              F32=-COS(TETAZ)*SIN(TETAY)*COS(TETAX)
```

```
      F3=F31+F32
C
C     MATRIX RD=R X PD
C
      RD11=R11*PD11+R12*PD21+R13*PD31
      RD21=R21*PD11+R22*PD21+R23*PD31
      RD31=R31*PD11+R32*PD21+R33*PD31
C
      RD12=R11*PD12+R12*PD22+R13*PD32
      RD22=R21*PD12+R22*PD22+R23*PD32
      RD32=R31*PD12+R32*PD22+R33*PD32
C
      RD13=R11*PD13+R12*PD23+R13*PD33
      RD23=R21*PD13+R22*PD23+R23*PD33
      RD33=R31*PD13+R32*PD23+R33*PD33
C
      E1NCX=RD11*F1+RD12*F2+RD13*F3
      E1NCY=RD21*F1+RD22*F2+RD23*F3
      E1NCZ=RD31*F1+RD32*F2+RD33*F3
C
C     IF E1NCX IS APPROACHING ZERO THEN ATAN IS 90 DEGREES
      IF(ABS(E1NCX).LT.0.000001)GO TO 1101
      BETA=ATAN(E1NCY/E1NCX)
      IF((E1NCY.GE.0).AND.(E1NCX.GT.0))TETHNC=BETA
      IF((E1NCY.GE.0).AND.(E1NCX.LT.0))TETHNC=3.14158+BETA
      IF((E1NCY.LT.0).AND.(E1NCX.GT.0))TETHNC=BETA
      IF((E1NCY.LT.0).AND.(E1NCX.LT.0))TETHNC=-(3.14158-BETA)
      GO TO 2201
1101  CONTINUE
      IF(E1NCY.GE.0)SIGN1=1.
      IF(E1NCY.LT.0)SIGN1=-1.
      TETHNC=SIGN1*1.57
2201  CONTINUE
C     TETHNP(I)=TETHNC - TETHNP(1)
C
C     MATRIX RH INVERSE
C
      RHI11=COS(TETAZ)*COS(TETAY)
      RHI12=-SIN(TETAZ)*COS(TETAY)
      RHI13=SIN(TETAY)
C
            RHI211=COS(TETAX)*SIN(TETAZ)
            RHI212=COS(TETAZ)*SIN(TETAY)*SIN(TETAX)
      RHI21=RHI211+RHI212
            RHI221=COS(TETAZ)*COS(TETAX)
            RHI222=-SIN(TETAZ)*SIN(TETAY)*SIN(TETAX)
      RHI22=RHI221+RHI222
      RHI23=-COS(TETAY)*SIN(TETAX)
C
            RHI311=SIN(TETAZ)*SIN(TETAX)
            RHI312=-COS(TETAZ)*SIN(TETAY)*COS(TETAX)
      RHI31=RHI311+RHI312
            RHI321=COS(TETAZ)*SIN(TETAX)
            RHI322=SIN(TETAZ)*SIN(TETAY)*COS(TETAX)
      RHI32=RHI321+RHI322
```

```
      RHI33=COS(TETAY)*COS(TETAX)
C
C     MATRIX PD x RHI
C
      RIDP11=PD11*RHI11+PD12*RHI21+PD13*RHI31
      RIDP12=PD11*RHI12+PD12*RHI22+PD13*RHI32
      RIDP13=PD11*RHI13+PD12*RHI23+PD13*RHI33
C
C     IF RIDP11 IS APPROACHING ZERO THEN ATAN IS 90 DEGREES
      IF(ABS(RIDP11).LT.0.000001) GO TO 2500
      BETA=ATAN(RIDP12/RIDP11)
      IF((RIDP12.GE.0).AND.(RIDP11.GT.0))PSIP=BETA
      IF((RIDP12.GE.0).AND.(RIDP11.LT.0))PSIP=3.14158+BETA
      IF((RIDP12.LT.0).AND.(RIDP11.GT.0))PSIP= BETA
      IF((RIDP12.LT.0).AND.(RIDP11.LT.0))PSIP=-(3.14158-BETA)
      GO TO 2600
 2500 CONTINUE
      IF(RIDP12.GE.0)SIGN1=1.
      IF(RIDP12.LT.0)SIGN1=-1.
      PSIP=SIGN1*1.57
 2600 CONTINUE
      TETHNP(I)=1.5708+PSIP
      IF(I.GT.1) GO TO 2601
      TETHN1=TETHNP(1)
      PSI1=PSI(1)
 2601 TETHNP(I)=TETHNP(I)-TETHN1
      PSI(I)=PSI(I)-PSI1
      GO TO 10
100   CONTINUE
C
      CALL FILEO(NAMTMP(7),RUNTMP,1,NUM,NERROR,RATIP)
      CALL FILEO(NAMTMP(8),RUNTMP,1,NUM,NERROR,ROTIP)
      CALL FILEO(NAMTMP(9),RUNTMP,5,NUM,NERROR,TENTYP)
      CALL FILEO(NAMTMP(10),RUNTMP,5,NUM,NERROR,TENTXP)
      CALL FILEO(NAMTMP(11),RUNTMP,5,NUM,NERROR,TETHNP)
      CALL FILEO(NAMTMP(12),RUNTMP,5,NUM,NERROR,PSI)
      CALL FILEO(NAMTMP(13),RUNTMP,6,NUM,NERROR,TOXOP)
      CALL FILEO(NAMTMP(14),RUNTMP,6,NUM,NERROR,TOYOP)
      CALL FILEO(NAMTMP(15),RUNTMP,6,NUM,NERROR,TOZOP)
      CALL FILEO(NAMTMP(16),RUNTMP,10,NUM,NERROR,FOXOP)
      CALL FILEO(NAMTMP(17),RUNTMP,10,NUM,NERROR,FOYOP)
      CALL FILEO(NAMTMP(18),RUNTMP,10,NUM,NERROR,FOZOP)
      CALL FILEO(NAMTMP(19),RUNTMP,6,NUM,NERROR,T1XLP)
      CALL FILEO(NAMTMP(20),RUNTMP,6,NUM,NERROR,T1YLP)
      CALL FILEO(NAMTMP(21),RUNTMP,6,NUM,NERROR,T1ZLP)
      CALL FILEO(NAMTMP(22),RUNTMP,6,NUM,NERROR,T1XTP)
      CALL FILEO(NAMTMP(23),RUNTMP,6,NUM,NERROR,T1YTP)
      CALL FILEO(NAMTMP(24),RUNTMP,6,NUM,NERROR,T1ZTP)
      RETURN
      END
*************************************************************************
*************************************************************************
      SUBROUTINE FILEO(NAMTMP,RUNTMP,UNTTMP,NUMBER,NERROR,ARRAY)
C
C
```

```
      BYTE UNITS(100)
      INTEGER*2 NUMB(100)
      CHARACTER*6 NAMTMP,RUNTMP,NAME(100),RUN(100)
      INTEGER UNTTMP,NUMBER,NERROR,NVAR
      REAL ARRAY(NUMBER),MIN(100),MAX(100)
C
C
      READ(1,REC=1)NVAR,NAME,RUN,MAX,MIN,UNITS,NUMB
      NVAR=NVAR+1
C
C
      MAX(NVAR)=-99999999.99
      MIN(NVAR)=99999999.99
C
      DO 1 I=2,NUMBER
      TMPMAX=AMAX1(ARRAY(I-1),ARRAY(I))
C
      MAX(NVAR)=AMAX1(MAX(NVAR),TMPMAX)
C
C
      TMPMIN=AMIN1(ARRAY(I-1),ARRAY(I))
C
      MIN(NVAR)=AMIN1(MIN(NVAR),TMPMIN)
 1    CONTINUE
C
C
      NAME(NVAR)=NAMTMP
      RUN(NVAR)=RUNTMP
      UNITS(NVAR)=UNTTMP
      NUMB(NVAR)=NUMBER
C
C
      WRITE(1,REC=1,ERR=900)NVAR,NAME,RUN,MAX,MIN,UNITS,NUMB

      WRITE(1,REC=(NVAR+1),ERR=900)(ARRAY(I),I=1,NUMBER)
C
      RETURN
900   NERROR=1
      RETURN
      END
```

SUBROUTINE TRQPHOR (HEAD)

```
C         TRQPHO.FOR
C
C         7/12/83.  J.B.  VARIABLE ANXXOS=ANXOP IS
C         IMPLEMEMTED/11 CHANGES
C
          BYTE UNITS(100)
          INTEGER*2 NUMB(100)
          INTEGER RECLEN,MREC,NVAR,NP(18),NERROR,RECNM(18),TYPE(2)
          REAL AAXXOS(600),AAYXOS(600),AAZXOS(600),PHAOXP(600),
     &      PHBO2P(600),PHCO3P(600),QHAOXS(600),QHBOXS(600),QHCOXS(600),
     &      RHAOXS(600),RHBOXS(600),RHCOXS(600),PNAOXP(600),PNBO2P(600),
     &      PNCO3P(600),ANXXOS(600),MIN(100),MAX(100),TARRAY(600),
     &      VNXXOS(600)
          CHARACTER*6 NAME(100),RUN(100),VRBLS(18),FILNM,NAMTMP

          PARAMETER (MREC=101)
          PARAMETER (RECLEN=598)

          COMMON/INDATA/TARRAY,PHAOXP,PHBO2P,PHCO3P,RHAOXS,RHBOXS,RHCOXS,
     &      QHAOXS,QHBOXS,QHCOXS,AAXXOS,AAYXOS,AAZXOS,PNAOXP,PNBO2P,
     &      PNCO3P,ANXXOS,VNXXOS
C
          DATA VRBLS/'PHAOXP','PHBO2P','PHCO3P','RHAOXS','RHBOXS',
     &      'RHCOXS','QHAOXS','QHBOXS','QHCOXS','AAXXOS','AAYXOS',
     &      'AAZXOS','PNAOXP','PNBO2P','PNCO3P','TIME','ANXXOS',
     &      'VNXXOS'/
C
        DATA TYPE/'OPEN','READ'/
C
C
C   STEP 1: OPEN INPUT FILE
C
C
        OPEN(UNIT=1,FILE='SCRTCH',RECL=RECLEN,STATUS='OLD',ERR=999,
     &     FORM='UNFORMATTED',ORGANIZATION='RELATIVE',ACCESS='DIRECT')
C
C
C   STEP 2: READ IN DIRECTORY
C
C
          READ(1,REC=1)NVAR,NAME,RUN,MAX,MIN,UNITS,NUMB
C
C
C
C   STEP 3: TEST FOR EXISTANCE OF VARIABLES
C
C
          DO 1 I=1,18
           DO 2 N=1,NVAR
             IF(NAME(N).NE.VRBLS(I))GO TO 3
         NP(I)=NUMB(N)
             RECNM(I)=N+1
             GO TO 1
  3           IF(N.EQ.NVAR)GO TO 900
  2        CONTINUE
  1       CONTINUE
C
C
C   STEP 4: READ APPROPRIATE RECORD
C
```

```
C
      READ(1,REC=RECNM(1))(PHAOXP(K),K=1,NP(1))
      READ(1,REC=RECNM(2))(PHBO2P(K),K=1,NP(2))
      READ(1,REC=RECNM(3))(PHCO3P(K),K=1,NP(3))
      READ(1,REC=RECNM(4))(RHAOXS(K),K=1,NP(4))
      READ(1,REC=RECNM(5))(RHBOXS(K),K=1,NP(5))
      READ(1,REC=RECNM(6))(RHCOXS(K),K=1,NP(6))
      READ(1,REC=RECNM(7))(QHAOXS(K),K=1,NP(7))
      READ(1,REC=RECNM(8))(QHBOXS(K),K=1,NP(8))
      READ(1,REC=RECNM(9))(QHCOXS(K),K=1,NP(9))
      READ(1,REC=RECNM(10))(AAXXOS(K),K=1,NP(10))
      READ(1,REC=RECNM(11))(AAYXOS(K),K=1,NP(11))
      READ(1,REC=RECNM(12))(AAZXOS(K),K=1,NP(12))
      READ(1,REC=RECNM(13))(PNAOXP(K),K=1,NP(13))
      READ(1,REC=RECNM(14))(PNBO2P(K),K=1,NP(14))
      READ(1,REC=RECNM(15))(PNCO3P(K),K=1,NP(15))
      READ(1,REC=RECNM(16))(TARRAY(K),K=1,NP(16))
       READ(1,REC=RECNM(17))(ANXXOS(K),K=1,NP(17))
       READ(1,REC=RECNM(18))(VNXXOS(K),K=1,NP(18))
C
 10   CONTINUE
      NERROR=0
C
C
      NAMTMP=RUN(RECNM(1)-1)
      NUM=NP(16)
      CALL TORQP(NERROR,NAMTMP,NUM)
C
      IF(NERROR.NE.0)GO TO 999
      GO TO 1000
C
C
C
C
C
C
C
C  ERROR MESSAGES
 900   WRITE(6,901)VRBLS(I)
 901   FORMAT(1X,'VARIABLE',1X,A10,1X,'IS NOT IN THE INPUT FILE',/,
     +       1X,'YOU ARE ABOUT TO BE RETURNED TO MONITOR LEVEL TO ',/,
     +       1X,'TRY AND RECTIFY THE PROBLEM',///)
C
C
C
      GO TO 1000
C
C
C
 999   WRITE(6,902)TYPE(NERROR)
C
 902   FORMAT(1X,A5,'ERROR ENCOUNTERED'/,
     +      1X,'PROGRAM HALT',///)
C
 1000  CONTINUE
C
      CLOSE(UNIT=1)
      END
*****************************************************************
*****************************************************************
```

```
      SUBROUTINE TORQP(NERROR,RUNTMP,NUM)
C
C     COMMON DATA FOR THE HEAD TORQUE PROGRAM (PHOTOGRAPHIC DATA)
C
      REAL AAXXOS(600),AAYXOS(600),AAZXOS(600),PHAOXP(600),
     &   PHBO2P(600),PHCO3P(600),QHAOXS(600),QHBOXS(600),QHCOXS(600),
     &   RHAOXS(600),RHBOXS(600),RHCOXS(600),PNAOXP(600),PNBO2P(600),
     &   PNCO3P(600),ANXXOS(600),TARRAY(600),VNXXOS(600)
      COMMON/INDATA/TARRAY,PHAOXP,PHBO2P,PHCO3P,RHAOXS,RHBOXS,RHCOXS,
     &   QHAOXS,QHBOXS,QHCOXS,AAXXOS,AAYXOS,AAZXOS,PNAOXP,PNBO2P,
     &   PNCO3P,ANXXOS,VNXXOS
C
C     4TH EDITION OF 2/22/83
C     FILE: TORQP3.FOR
C
C     COMPUTATION OF MOMENTS AND FORCES AT THE OCCIPITAL CONDYLES
C     DUE TO HEAD DECELERATION AND GRAVITY AS DERIVED FROM
C     PHOTOGRAPHIC DATA
C
C     PROGRAM CONSTANTS -- DIFFERENT FOR EACH SUBJECT #
C
C     MH=MASS OF HEAD
C     G=ACCELERATION OF GRAVITY AT NBDL
C     RGAX = THE COMPONENT OF LINEAR POSITION OF THE
C            HEAD CENTER OF GRAVITY ALONG THE X-AXIS
C            OF THE HEAD ANATOMICAL COORDINATE SYSTEM.
C     RGAZ = SAME AS ABOVE EXCEPT FOR  THE Z-AXIS.
C     RGOX = THE COMPONENT OF LINEAR POSITION OF THE
C            HEAD CENTER OF GRAVITY ALONG THE X-AXIS
C            OF THE HEAD ANATOMICAL COORDINATE SYSTEM MEAURED
C            FROM THE OCCIPITAL CONDYLES.
C     RGOZ = SAME AS ABOVE EXCEPT FOR THE Z-AXIS.
C     IX,IY,IZ = THE COMPONENT OF CENTROIDAL MASS MOMENT
C            OF INERTIA OF THE INSTRUMENTED HEAD ABOUT AN
C            AXIS PARALLEL TO THE X, Y, Z-AXIS OF THE
C            HEAD ANATOMICAL SYSTEM, RESPECTIVELY.
C     PXY = THE COMPONENT OF CENTROIDAL MASS PRODUCT OF INERTIA
C            OF THE INSTRUMENTED HEAD ABOUT AN AXIS PARALLEL
C            TO EITHER THE XOR Y AXIS OF THE HEAD
C            ANATOMICAL COORD. SYSTEM AND DEFINED BY AN
C            INTEGRAL OF XYD(MH).
C     PXZ,PYX,PYZ,PZX,PZY = SAME AS ABOVE EXCEPT  FOR
C            THE RESPECTIVE AXES.
C
C     PROGRAM VARIABLES (ARRAYS)
C
C     PHAOXP=ANGLE ROTATION OF THE HEAD ABOUT THE X AXIS OF
C            THE HEAD ANATOMICAL COORD. SYSTEM AS DERIVED FROM
C            PHOTOGRAPHIC DATA
C     PHBO2P=SAME AS ABOVE EXCEPT FOR THE Y AXIS
C     PHCO3P=SAME AS ABOVE EXCEPT FOR THE Z AXIS
C     RHAOXS=ANGULAR VELOCITY OF THE HEAD ABOUT THE
C            X AXIS OF THE HEAD ANATOMICAL COORD.
C            SYSTEM AS DERIVED FROM ACCELEROMETER
C            DATA
C     RHBOXS=SAME AS ABOVE EXCEPT FOR THE Y AXIS
C     RHCOXS=SAME AS ABOVE EXCEPT FOR THE Z AXIS
C     QHAOXS=ANGULAR ACCELERATION OF THE HEAD
C             ABOUT THE X AXIS OF HEAD ANATOMICAL
C             COORD.  SYSTEM AS DERIVED FROM
```

```
C               ACCELEROMETER  DATA
C       GHBOXS=SAME AS ABOVE EXCEPT FOR THE Y AXIS
C       GHCOXS=SAME AS ABOVE EXCEPT FOR THE Z AXIS
C       AAXXOS=THE X COMPONENT OF ACCELERATION OF
C               THE HEAD ANATOMICAL ORIGIN (THE
C               LABORATORY COORD. SYSTEM) WITH
C               RESPECT TO THE FIXED LABORATORY
C               COORD. SYSTEM AS DERIVED FROM
C               ACCELEROMETER DATA
C       AAYXOS=SAME AS ABOVE EXCEPT ABOUT THE
C               Y AXIS
C       AAZXOS=SAME AS ABOVE EXCEPT ABOUT THE
C               Z AXIS
C       PNAOXP=ANGLE OF ROTATION OF THE HEAD ABOUT THE
C               X AXIS OF THE HEAD ANATOMICAL COORD.
C               SYSTEM (INITIALLY ALIGNED WITH THE LABORATORY
C               COORD. SYSTEM) AS DERIVED FROM ACCELEROMETER
C               DATA
C       PNBO2P=SAME AS ABOVE EXCEPT ABOUT THE CARRIED X AXIS
C       PNCO3P=SAME AS ABOVE EXCEPT ABOUT THE CARRIED Z AXIS
C       TARRAY=TIME MARKS OF DATA POINTS RECORDING
C       ANXXOS=THE COMPONENT OF LINEAR ACCELERATION
C               OF THE T1 ANATOMICAL ORIGIN ALONG THE
C               X-AXIS OF THE LABORATORY COORDINATE
C               SYSTEM WITH RESPECT TO THE FIXED
C               LABORATORY COORDINATE SYSTEM AS DERIVED
C               FROM T1 MOUNT ACCELEROMETER DATA
C       VNXXOS=THE COMPONENT OF LINEAR VELOCITY OF THE
C               T1 ANATOMICAL ORIGIN ALONG THE
C               X-AXIS OF THE LABORATORY COORDINATE SYSTEM
C               WITH RESPECT TO THE FIXEDLABORATORYCOORDINAT
C               SYSTEMAS DERIVED FROM T1 MOUNT ACCELEROMETER
C               DATA.
C
C       OUTPUT VARIABLES (ARRAYS)
C
C       AGXP=X-COMPONENT OF ACCELERATION OF THE HEAD C.G.  (THE HEAD
C               ANATOMICAL C.S.) WRT. THE LABORATORY COORD. SYSTEM
C               AS DERIVED FROM PHOTOGRAPHIC DATA
C       AGYP,AGZP=Y- AND Z-COORDINATES OF THE ABOVE
C       TOXP=THE COMPONENT OF MOMENT APPLIED BY THE NECK TO THE
C               HEAD ABOUT AN AXIS PARALLEL TO THE X AXIS OF THE
C               HEAD ANATOMICAL COORD. SYSTEM AS DERIVED FROM
C               PHOTOGRAPHIC DATA
C       TOYP,TOZP=SAME ABOUT THE Y- AND Z-AXES
C       FOXP=THE COMPONENT OF FORCE APPLIED BY THE NECK TO THE
C               HEAD AT THE OCCIPITAL CONDYLE PARALLEL TO
C               THE X AXIS OF HEAD ANATOMICAL COORD.
C               SYSTEM AS DERIVED FROM PHOTOGRAPHIC DATA
C       FOYP,FOZP=SAME ABOUT THE Y AND Z AXES
C       THTIYP=THE ANGLE OF ROTATION OF A PLANE FORMED
C               BY THE Y AXIS OF THE T1 ANATOMICAL
C               COORD. SYSTEM AND A UNIT VECTOR ALONG
C               THE Z AXIS OF THE HEAD ANATOMICAL COORD.
C               SYSTEM WITH RESPECT TO THE PLANE FORMED
C               BY THE Y AND Z AXES OF THE T1 ANATOMICAL
C               COORD. SYSTEM
C       THTIXP=THE ANGLE OF ROTATION OF A PLANE FORMED
C               BY THE X-AXIS OF THE T1 ANATOMICAL COORD.
C               SYSTEM AND A UNIT VECTOR ALONG THE Z
```

```
C                AXIS OF THE HEAD ANATOMICAL COORD. SYSTEM
C                WITH RESPECT TO THE PLANE FORMED BY THE
C                X AND Z AXES OF THE T1 ANATOMICAL COORD.
C                SYSTEM
C        TOXLP=THE COMPONENT OF MOMENT APPLIED BY THE NECK
C                TO THE HEAD ABOUT AN AXIS PARALLEL TO THE
C                LABORATORY X-AXIS AND PASSING THROUGH THE ORIGIN
C                OF THE OCCIPITAL COORDINATE SYSTEM
C        TOYLP=THE COMPONENT OF MOMENT APPLIED BY THE NECK TO THE
C                HEAD ABOUT AN AXIS PARALLEL TOTHE LABORATORY
C                Y-AXIS AND PASSING THROUGH THE ORIGIN OF THE
C                OCCIPITAL COORDINATE SYSTEM
C        TOZLP=THE COMPONENT OF MOMENT APPLIED BY THE NECK
C                TO THE HEAD ABOUT AN AXIS PARALLEL TO THE LABORATORY
C                Z-AXIS AND PASSING THROUGH THE ORIGIN OF THE NECK
C                CHORD COORDINATE SYSTEM
C        FOXLP=THE COMPONENT OF FORCE APPLIED BY THE NECK TO THE
C                HEAD PARALLEL TO THE LABORATORY X-AXIS AND PASSING
C                THROUGH THE ORIGIN OF THE OCCIPITAL COORDINATE
C                SYSTEM
C        FOYLP=THE COMPONENT OF FORCE APPLIED BY THE NECK TO THE
C                HEAD PARALLEL TO THE LABORATORY Y-AXIS AND PASSING
C                THROUGH THE ORIGIN OF THE OCCIPITAL COORDINATE
C                SYSTEM
C        FOZLP=THE COMPONENT OF FORCE APPLIED BY THE NECK TO THE
C                HEAD PARALLEL TO THE LABORATORY Z-AXIS AND
C                PASSING THROUGH THE ORIGIN OF THE OCCIPITAL
C                COORDINATE SYSTEM
C        TOXTP=THE COMPONENT OF MOMENT APPLIED BY THE
C                NECK TO THE HEAD ABOUT THE X-AXIS OF THE T10
C                COORDINATE SYSTEM
C        TOYTP=THE COMPONENT OF MOMENT APPLIED BY THE NECK TO TT
C                THE HEAD ABOUT THE Y-AXIS BY THE T10 COORDINATE
C                SYSTEM
C        TOZTP=THE COMPONENT OF MOMENT APPLIED BY THE NECK TO THE
C                HEAD ABOUT THE Z-AXIS OF THE T10 COORDINATE
C                SYSTEM
C        FOXTP=THE COMPONENT OF FORCE APPLIED BY NECK TO THE
C                HEAD ALONG THE X-AXIS OF THE T10 COORDINATE
C                SYSTEM
C        FOYTP=THE COMPONENT OF FORCE APPLIED BY THE NECK TO
C                HEAD ALONG THE Y-AXIS OF THE T10 COORDINATE
C                SYSTEM
C        FOZTP=THE COMPONENT OF FORCE APPLIED BY THE NECK TO THE
C                HEAD ALONG THE Z-AXIS OF THE T10 COORDINATE SYSTEM
C        ANXOP=ANXXOS
C        VNXOP=VNXXOS
C
         REAL AGXP(600),AGYP(600),AGZP(600),TOXP(600),TOYP(600),
     &     TOZP(600),FOXP(600),FOYP(600),FOZP(600),THTIYP(600),
     &     THTIXP(600),TOXLP(600),TOYLP(600),TOZLP(600),FOXLP(600),
     &     FOYLP(600),FOZLP(600),TOXTP(600),TOYTP(600),TOZTP(600),
     &     FOXTP(600),FOYTP(600),FOZTP(600),ANXOP(600),VNXOP(600)
         CHARACTER*6 NAMTMP(25),RUNTMP
         INTEGER NERROR
         REAL IX,IY,IZ,MH
C
         DATA NAMTMP/'TOXP','TOYP','TOZP','THTIYP','THTIXP','AGXP',
     &     'AGYP','AGZP','FOXP','FOYP','FOZP','TOXLP','TOYLP','TOZLP',
     &     'FOXLP','FOYLP','FOZLP','TOXTP','TOYTP','TOZTP','FOXTP',
```

```
     &     'FOYTP', 'FOZTP', 'ANXOP', 'VNXOP'/
C
C
C         SUBJECT NUMBER SELECTION
1700      CONTINUE
          READ(5,*)NJCT
          WRITE(6,1755)NJCT
1755      FORMAT(5X,'SUBJECT NUMBER=',I5)
          IF(NJCT.EQ.1)GO TO 1001
          IF(NJCT.EQ.83)GO TO 1083
          IF(NJCT.EQ.93)GO TO 1093
          IF(NJCT.EQ.96)GO TO 1096
          IF(NJCT.EQ.44)GOTO 1044
          IF(NJCT.EQ.64)GOTO 1064
          IF(NJCT.EQ.65)GOTO 1065
          IF(NJCT.EQ.67)GOTO 1067
1701      CONTINUE
          WRITE(6,1702)
1702      FORMAT(1X,'INCORRECT SUBJECT NUMBER')
          STOP
C
1001      CONTINUE
          MH=4.6
          RGAX=0.012
          RGAZ=0.029
          RGOX=0.0234
          RGOZ=0.055
          IX=0.0215
          IY=0.0278
          IZ=0.0179
          PXY=-0.000
          PYX=-0.000
          PXZ=-0.0057
          PZX=-0.0057
          PYZ=0.000
          PZY=0.000
          RGOY=0.0
          GO TO 999
C
1083      CONTINUE
          MH=4.532
          RGAX=0.012
          RGAZ=0.029
          RGOX=0.023
          RGOZ=0.055
          IX=0.0211
          IY=0.0261
          IZ=0.0174
          PXY=0
          PYX=0.
          PXZ=-.0056
          PZX=-.0056
          PYZ=0.
          PZY=0.
          RGOY=0.0
          GO TO 999
C
1093      CONTINUE
          MH=4.03
          RGAX=0.012
```

```
      RGAZ=0. 029
      RGOX=0. 023
      RGOZ=0. 055
      IX=0. 0174
      IY=0. 0215
      IZ=0. 0142
      PXY=0.
      PYX=0.
      PXZ=-0. 0049
      PZX=-0. 0049
      PYZ=0.
      PZY=0.
      RGOY=0.
      GO TO 999
C
1044  CONTINUE
      MH=4. 37
      RGAX=0. 012
      RGAZ=. 029
      RGOX=. 023
      RGOZ=. 055
      IX=. 0200
      IY=. 0258
      IZ=. 0166
      PXY=0. 0
      PYX=0. 0
      PXZ=-. 0053
      PZX=-. 0053
      PYZ=0. 0
      PZY=0. 0
      RGOY=0. 0
      GO TO 999
C
1064  CONTINUE
      MH=4. 90
      RGAX=. 012
      RGAZ=. 029
      RGOX=. 023
      RGOY=0.
      RGOZ=. 055
      IX=. 0236
      IY=. 0306
      IZ=. 0198
      PXY=0.
      PYX=0.
      PXZ=-. 0058
      PZX=-. 0058
      PYZ=0.
      PZY=0.
      GOTO 999
C
1065  CONTINUE
      MH=5. 11
      RGAX=. 012
      RGAZ=. 029
      RGOX=. 023
      RGOY=0.
      RGOZ=. 055
      IX=. 0250
      IY=. 0335
```

```
          IZ=.0211
          PXY=0.
          PYX=0.
          PXZ=-.0060
          PZX=-.0060
          PYZ=0.
          PZY=0.
          GOTO 999
C
1067      CONTINUE
          MH=4.66
          RGAX=.012
          RGAZ=.029
          RGOX=.023
          RGOY=0.
          RGOZ=.055
          IX=.0220
          IY=.0290
          IZ=.0184
          PXY=0.
          PYX=0.
          PXZ=-.0057
          PZX=-.0057
          PYZ=0.
          PZY=0.
          GOTO 999
C
1096      CONTINUE
          MH=0.
          RGAX=0.012
          RGAZ=0.029
          RGOX=0.023
          RGOZ=0.055
          IX=0
          IY=0.
          IZ=0.
          PXY=0.
          PYX=0.
          PXZ=0.
          PZX=0.
          PYZ=0.
          PZY=0.
          RGOY=0.
          GO TO 999
C
1725      CONTINUE
C
999       CONTINUE
          DELT=0.0005
          G=9.81
C
        TPMAX=TARRAY(NUM)
        TSMAX=DELT*598
        IF(TPMAX.LE.TSMAX)GO TO 55
        DO 11 K=1,NUM
        K1=K
        TAR=TARRAY(K1)
        IF(TAR.LE.TSMAX)GO TO 11
        KC=K
        KMAX=KC-1
```

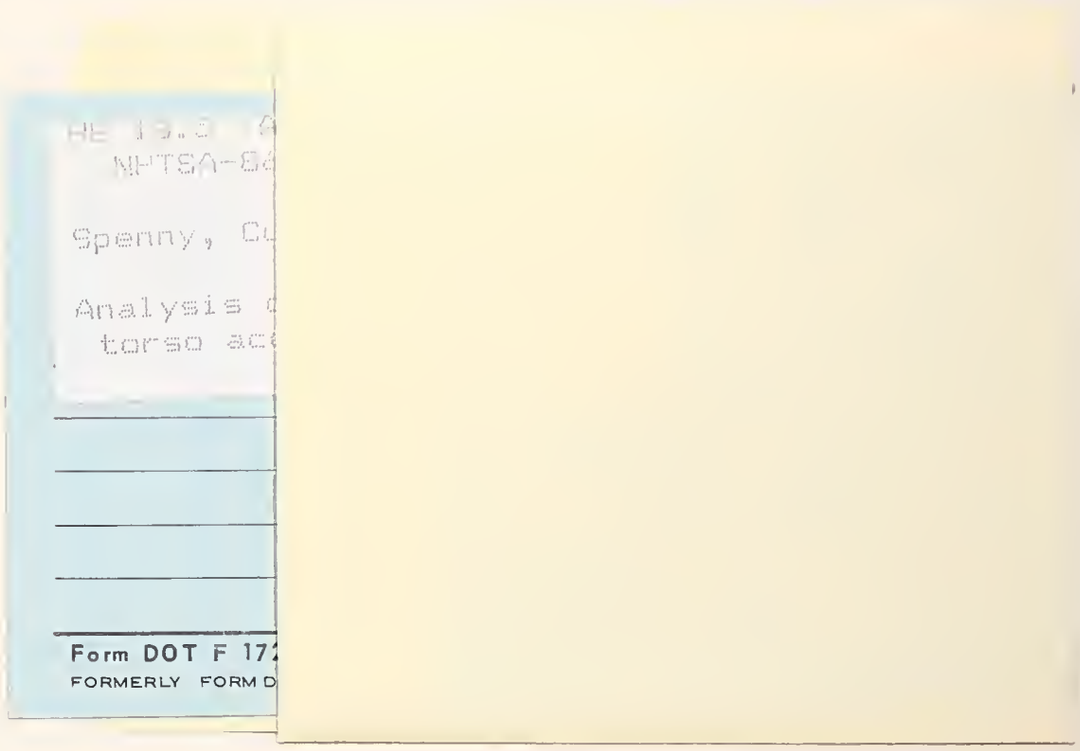
```
      GO TO 22
C
 11   CONTINUE
 22   CONTINUE
C
      NUM=KMAX
C
55     CONTINUE
C
       TET1XO=PNAOXP(1)
       TET1YO=PNBO2P(1)
       TET1ZO=PNCO3P(1)
C
C      MATRIX P -- FOR EQUATIONS (19)
C
       P11=COS(TET1ZO)*COS(TET1YO)
              P211=COS(TET1XO)*SIN(TET1ZO)
              P212=COS(TET1ZO)*SIN(TET1YO)*SIN(TET1XO)
       P21=P211+P212
              P311=SIN(TET1ZO)*SIN(TET1XO)
              P312=-COS(TET1ZO)*SIN(TET1YO)*COS(TET1XO)
       P31=P311+P312
       P12=-SIN(TET1ZO)*COS(TET1YO)
              P221=COS(TET1ZO)*COS(TET1XO)
              P222=-SIN(TET1ZO)*SIN(TET1YO)*SIN(TET1XO)
       P22=P221+P222
              P321=COS(TET1ZO)*SIN(TET1XO)
              P322=SIN(TET1ZO)*SIN(TET1YO)*COS(TET1XO)
       P32=P321+P322
       P13=SIN(TET1YO)
       P23=-COS(TET1YO)*SIN(TET1XO)
       P33=COS(TET1YO)*COS(TET1XO)
C
C
       I=0
C
 10    CONTINUE
       I=I+1
       IF(I.EQ.(NUM+1))GO TO 100
       TAR=TARRAY(I)
       DO 33 J=1,598
        JCOPY=J
        TIME=DELT*JCOPY
       IF(TIME.LT.TAR)GO TO 33
C
C
C
C
        I2=JCOPY
        GO TO 44
33      CONTINUE
44      CONTINUE
        I1=I2-1
C       CALCULATION OF THE NEEDED VARIABLES
C       THROUGH INTERPOLATION OF ACCELEROMETER
C       DATA
C
       TAR=TARRAY(I)
       CF=(TAR-DELT*I1)/DELT
       FC=1.-CF
```

```
      TETAX=PHAOXP(I)
      TETAY=PHBO2P(I)
      TETAZ=PHCO3P(I)
      ALFX=FC*QHAOXS(I1)+CF*QHAOXS(I2)
      ALFY=FC*QHBOXS(I1)+CF*QHBOXS(I2)
      ALFZ=FC*QHCOXS(I1)+CF*QHCOXS(I2)
      AACX=FC*AAXXOS(I1)+CF*AAXXOS(I2)
      AACY=FC*AAYXOS(I1)+CF*AAYXOS(I2)
      AACZ=FC*AAZXOS(I1)+CF*AAZXOS(I2)
      WX=FC*RHAOXS(I1)+CF*RHAOXS(I2)
      WY=FC*RHBOXS(I1)+CF*RHBOXS(I2)
      WZ=FC*RHCOXS(I1)+CF*RHCOXS(I2)
      ANXOP(I)=FC*ANXXOS(I1)+CF*ANXXOS(I2)
      VNXOP(I)=FC*VNXXOS(I1)+CF*VNXXOS(I2)
C
C     EQUATIONS (13)
C     **************
C
            G11=-SIN(TETAZ)*SIN(TETAX)
            G12=COS(TETAZ)*SIN(TETAY)*COS(TETAX)
      GX=(G11+G12)*G
            G21=-COS(TETAZ)*SIN(TETAX)
      G22=-SIN(TETAZ)*SIN(TETAY)*COS(TETAX)
      GY=(G21+G22)*G
      GZ=(-COS(TETAY)*COS(TETAX))*G
C
C     EQUATIONS (17)
C     **************
C
      A11=COS(TETAZ)*COS(TETAY)
            A121=COS(TETAX)*SIN(TETAZ)
            A122=COS(TETAZ)*SIN(TETAY)*SIN(TETAX)
      A12=A121+A122
            A131=SIN(TETAZ)*SIN(TETAX)
            A132=-COS(TETAZ)*SIN(TETAY)*COS(TETAX)
      A13=A131+A132
      A21=-SIN(TETAZ)*COS(TETAY)
            A221=COS(TETAZ)*COS(TETAX)
            A222=-SIN(TETAZ)*SIN(TETAY)*SIN(TETAX)
      A22=A221+A222
            A231=COS(TETAZ)*SIN(TETAX)
            A232=SIN(TETAZ)*SIN(TETAY)*COS(TETAX)
      A23=A231+A232
      A31=SIN(TETAY)
      A32=-COS(TETAY)*SIN(TETAX)
      A33=COS(TETAY)*COS(TETAX)
      AAX=A11*AACX+A12*AACY+A13*AACZ
      AAY=A21*AACX+A22*AACY+A23*AACZ
      AAZ=A31*AACX+A32*AACY+A33*AACZ
C
C     EQUATIONS (18)
C     **************
C
            B11=-(WZ**2+WY**2)*RGAX
            B12=WX*WZ*RGAZ
      B1=B11+B12
            B21=WX*WZ*RGAX
            B22=WY*WZ*RGAZ
      B2=B21+B22
            B31=WX*WZ*RGAX
```

```
                B32=-(WX**2+WY**2)*RGAZ
        B3=B31+B32
        C1=RGAZ*ALFY
                C21=RGAX*ALFZ
                C22=-RGAZ*ALFX
        C2=C21+C22
        C3=-RGAX*ALFY
        AGX=AAX+B1+C1
        AGY=AAY+B2+C2
        AGZ=AAZ+B3+C3
C
C
C       EQUATIONS (15)
C
        T11=IX*ALFX
        T12=IY*ALFY
        T13=IZ*ALFZ
        T21=(IZ-IY)*WY*WZ
        T22=(IX-IZ)*WX*WZ
        T23=(IY-IX)*WX*WY
                T311=MH*RGOZ*(GY-AGY)
                T312=-MH*RGOY*(GY-AGZ)
        T31=T311+T312
                T321=-MH*RGOZ*(GX-AGX)
                T322=MH*RGOX*(GZ-AGZ)
        T32=T321+T322
                T331=-MH*RGOX*(GY-AGY)
                T332=MH*RGOY*(GX-AGX)
        T33=T331+T332
        T41=-PXY*ALFY-PXZ*ALFZ
        T42=-PYX*ALFX-PYZ*ALFZ
        T43=-PZX*ALFX-PZY*ALFY
                T511=PYX*WX*WZ
                T512=PYZ*(WZ**2)
                T513=-PZX*WX*WY
                T514=-PZY*(WY**2)
        T51=T511+T512+T513+T514
                T521=-PXY*WY*WZ
                T522=-PXZ*(WZ**2)
                T523=PZX*(WX**2)
                T524=PZY*WX*WY
        T52=T521+T522+T523+T524
                T531=PXY*(WY**2)
                T532=PXZ*WY*WZ
                T533=-PYX*(WX**2)
                T534=-PYZ*WX*WZ
        T53=T531+T532+T533+T534
        TOX=T11+T21+T31+T41+T51
        TOY=T12+T22+T32+T42+T52
        TOZ=T13+T23+T33+T43+T53
C
C       EQUATIONS (16)
C       ***************
C
        FOX=MH*(AGX-GX)
        FOY=MH*(AGY-GY)
        FOZ=MH*(AGZ-GZ)
C
C       EQUATIONS (22)
C       ***************
```

```
C
      TOXL=TOX*A11+TOY*A21+TOZ*A31
      TOYL=TOX*A12+TOY*A22+TOZ*A32
      TOZL=TOX*A13+TOY*A23+TOZ*A33
C
C     EQUATIONS (23)
C     **************
C
      FOXL=FOX*A11+FOY*A21+FOZ*A31
      FOYL=FOX*A12+FOY*A22+FOZ*A32
      FOZL=FOX*A13+FOY*A23+FOZ*A33
C     OUTPUT VARIABLES (ARRAYS)
C
      AGXP(I)=AGX
      AGYP(I)=AGY
      AGZP(I)=AGZ
      TOXP(I)=TOX
      TOYP(I)=TOY
      TOZP(I)=TOZ
      FOXP(I)=FOX
      FOYP(I)=FOY
      FOZP(I)=FOZ
      TOXLP(I)=TOXL
      TOYLP(I)=TOYL
      TOZLP(I)=TOZL
      FOXLP(I)=FOXL
      FOYLP(I)=FOYL
      FOZLP(I)=FOZL
C     EQUATIONS (19)
C     **************
C
      TETA1X=PNAOXP(I)
      TETA1Y=PNBO2P(I)
      TETA1Z=PNCO3P(I)
C
      D11=COS(TETA1Z)*COS(TETA1Y)
      D21=-SIN(TETA1Z)*COS(TETA1Y)
      D31=SIN(TETA1Y)
            D121=COS(TETA1X)*SIN(TETA1Z)
            D122=COS(TETA1Z)*SIN(TETA1Y)*SIN(TETA1X)
      D12=D121+D122
            D221=COS(TETA1Z)*COS(TETA1X)
            D222=-SIN(TETA1Z)*SIN(TETA1Y)*SIN(TETA1X)
      D22=D221+D222
      D32=-COS(TETA1Y)*SIN(TETA1X)
            D131=SIN(TETA1Z)*SIN(TETA1X)
            D132=-COS(TETA1Z)*SIN(TETA1Y)*COS(TETA1X)
      D13=D131+D132
            D231=COS(TETA1Z)*SIN(TETA1X)
            D232=SIN(TETA1Z)*SIN(TETA1Y)*COS(TETA1X)
      D23=D231+D232
      D33=COS(TETA1Y)*COS(TETA1X)
C
      F1=SIN(TETAY)
      F2=-COS(TETAY)*SIN(TETAX)
      F3=COS(TETAY)*COS(TETAX)
C
C     MATRIX PD=P X D
C
      PD11=(P11*D11)+(P12*D21)+(P13*D31)
```

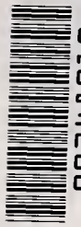
```
      MIN(NVAR)=99999999.99
C
      DO 1 I=2,NUMBER
      TMPMAX=AMAX1(ARRAY(I-1),ARRAY(I))
C
      MAX(NVAR)=AMAX1(MAX(NVAR),TMPMAX)
C
C
      TMPMIN=AMIN1(ARRAY(I-1),ARRAY(I))
C
      MIN(NVAR)=AMIN1(MIN(NVAR),TMPMIN)
 1    CONTINUE
C
C
      NAME(NVAR)=NAMTMP
      RUN(NVAR)=RUNTMP
      UNITS(NVAR)=UNTTMP
      NUMB(NVAR)=NUMBER
C     .
C
      WRITE(1,REC=1,ERR=900)NVAR,NAME,RUN,MAX,MIN,UNITS,NUMB

      WRITE(1,REC=(NVAR+1),ERR=900)(ARRAY(I),I=1,NUMBER)
C
      RETURN
900   NERROR=1
      RETURN
      END
```


HE 18.5
NHTSA-8
Spenny, C
Analysis
torso ac
Form DOT F 17
FORMERLY FORM D

U.S. Department
Transportation

**search and
ecial Programs
Iministration**

idall Square
nbridge, Massachusetts 02142



Official Business
Penalty for Private Use $300

Postage and Fees Paid
Research and Special
Programs Administration
DOT 513